



Seafile

Sharing & Collaboration

目錄

| | |
|----------------------------|---------|
| Introduction | 1.1 |
| 概覽 | 1.2 |
| Seafile 组件 | 1.2.1 |
| 常见问题解答 | 1.2.2 |
| 我要参与 | 1.2.3 |
| Linux 下部署 Seafile 服务器 | 1.3 |
| 部署 Seafile 服务器（使用 SQLite） | 1.3.1 |
| 部署 Seafile 服务器（使用 MySQL） | 1.3.2 |
| Nginx 下配置 Seahub | 1.3.3 |
| Nginx 下启用 Hhttps | 1.3.4 |
| Apache 下配置 Seahub | 1.3.5 |
| Apache 下启用 Hhttps | 1.3.6 |
| 升级 | 1.3.7 |
| 其他部署说明 | 1.3.8 |
| 开机启动 Seafile | 1.3.8.1 |
| 防火墙设置 | 1.3.8.2 |
| Logrotate 管理系统日志 | 1.3.8.3 |
| 使用 Memcached | 1.3.8.4 |
| 使用 NAT | 1.3.8.5 |
| 从 SQLite 迁移至 MySQL | 1.3.8.6 |
| 安装常见问题 | 1.3.9 |
| Seafile 服务器 5.0 中配置文件新位置 | 1.3.10 |
| Windows 下部署 Seafile 服务器 | 1.4 |
| 下载安装 Windows 版 Seafile 服务器 | 1.4.1 |
| 安装 Seafile 为 Windows 服务 | 1.4.2 |
| 所用端口说明 | 1.4.3 |
| 升级 | 1.4.4 |
| 从 Windows 迁移到 Linux | 1.4.5 |
| 垃圾回收 | 1.4.6 |

| | |
|----------------------|----------|
| 部署 Seafile 专业版服务器 | 1.5 |
| 下载安装 Seafile 专业版服务器 | 1.5.1 |
| 从社区版迁移至专业版 | 1.5.2 |
| 升级 | 1.5.3 |
| 使用 Oracle 数据库部署 | 1.5.4 |
| AD 集成 | 1.5.5 |
| 专业版 AD/LDAP 配置 | 1.5.5.1 |
| 同步 AD 群组 | 1.5.5.2 |
| 配置 ADFS 登陆 | 1.5.5.3 |
| Office 文件预览和编辑 | 1.5.6 |
| 开启 Office/PDF 文件在线预览 | 1.5.6.1 |
| Office Web App 集成 | 1.5.6.2 |
| 文件搜索说明 | 1.5.7 |
| 病毒扫描 | 1.5.8 |
| 卡巴斯基病毒扫描配置 | 1.5.8.1 |
| Web 文件断点续传 | 1.5.9 |
| 存储后端 | 1.5.10 |
| Amazon S3 下安装 | 1.5.10.1 |
| Ceph 下安装 | 1.5.10.2 |
| 使用阿里云OSS存储 | 1.5.10.3 |
| 集群部署 | 1.5.11 |
| 集群部署 | 1.5.11.1 |
| MariaDB/Memcached 集群 | 1.5.11.2 |
| 后台任务节点高可用 | 1.5.11.3 |
| HAProxy高可用 | 1.5.11.4 |
| HAProxy HTTPS 配置 | 1.5.11.5 |
| NFS 下集群安装 | 1.5.11.6 |
| 升级集群 | 1.5.11.7 |
| 用户角色与权限 | 1.5.12 |
| 高级认证 | 1.5.13 |
| 两步认证 | 1.5.13.1 |
| 高级维护工具 | 1.5.14 |

| | |
|------------------------------------|----------|
| 向seafile中导入目录 | 1.5.14.1 |
| 配置选项 | 1.5.15 |
| 常见问题解答 | 1.5.16 |
| 软件许可协议 | 1.5.17 |
| 服务器个性化配置 | 1.6 |
| ccnet.conf | 1.6.1 |
| seafile.conf | 1.6.2 |
| seahub_settings.py | 1.6.3 |
| 发送邮件提醒 | 1.6.4 |
| 个性化邮件提醒 | 1.6.5 |
| 存储容量与文件上传/下载大小限制 | 1.6.6 |
| 自定义 Web | 1.6.7 |
| 日常维护 | 1.7 |
| 账户管理 | 1.7.1 |
| 日志 | 1.7.2 |
| 清理数据库 | 1.7.3 |
| 备份与恢复 | 1.7.4 |
| Seafile FSCK | 1.7.5 |
| Seafile GC | 1.7.6 |
| WebDAV 和 FUSE 扩展 | 1.8 |
| WebDAV 扩展 | 1.8.1 |
| FUSE 扩展 | 1.8.2 |
| 安全选项 | 1.9 |
| 安全特性 | 1.9.1 |
| 日志和审计 | 1.9.2 |
| 开发文档 | 1.10 |

简介

Seafile 是一个开源的文件云存储平台，解决文件集中存储、同步、多平台访问的问题，注重安全和性能。

Seafile 通过“资料库”来分类管理文件，每个资料库可单独同步，用户可加密资料库，且密码不会保存在服务器端，所以即使是服务器管理员也无权访问你的文件。

Seafile 允许用户创建“群组”，在群组内共享和同步文件，方便了团队协作工作。

软件许可协议

Seafile 及其桌面、移动客户端遵循 GPLv3。

Seahub（Seafile 服务器的 web 端）遵循 Apache License。

关于

本手册“源码”托管在Github <https://github.com/haiwen/seafile-docs-cn>

联系方式

- Twitter: @seafile <https://twitter.com/seafile>

更多内容

- 更多内容请见 [Seafile Wiki](#)

概览

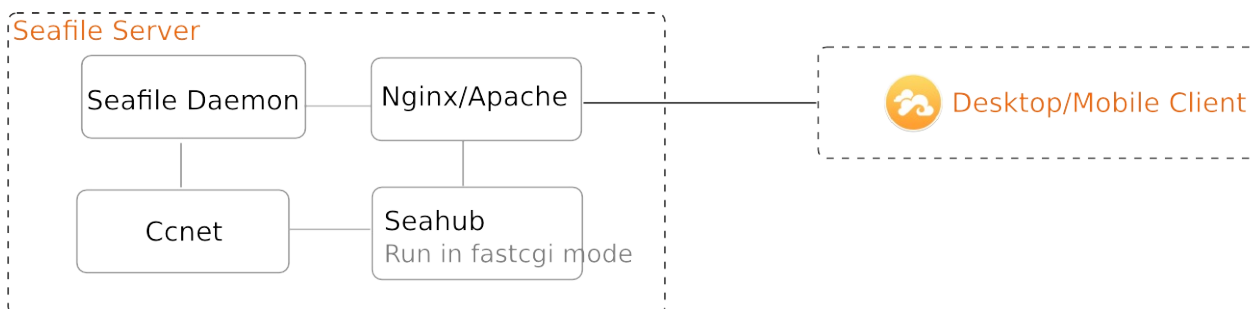
- [Seafile 组件](#)
- [FAQ](#)
- [Contribution](#)

Seafile服务器组件

Seafile 包含以下系统组件：

- **Seahub**：网站界面，供用户管理自己在服务器上的数据和账户信息。Seafile 服务器通过"unicorn"（一个轻量级的Python HTTP服务器）来提供网站支持。Seahub作为unicorn的一个应用程序来运行。
- **Seafile server** (`seaf-server`)：数据服务进程, 处理原始文件的上传/下载/同步。
- **Ccnet server** (`ccnet-server`)：内部 RPC 服务进程，连接多个组件。
- **Controller**: 监控 ccnet 和 seafile 进程，必要时会重启进程。

下面这张图显示了将 Seafile 部署在 Nginx/Apache 后的架构。



- 所有 Seafile 服务都可以配置在 Nginx/Apache 后面，由 Nginx/Apache 提供标准的 http(s) 访问。
- 当用户通过 seahub 访问数据时，seahub 通过 ccnet 提供的内部 RPC 来从 seafile server 获取数据。

FAQ

无法在网页上下载/上传文件

请检查下 `SERVICE_URL` 和 `FILE_SERVER_ROOT` 这两个配置选项是否正确设置。如果使用内置的 Web 服务器，监听在 8000 端口上，应该是

```
SERVICE_URL = http://IP:8000
FILE_SERVER_ROOT 选项不用配置
```

如果是使用 Nginx/Apache 为 Web 服务器 (确保 Seahub 以 factcgi 模式来运行)，应该是

```
SERVICE_URL = http://IP
FILE_SERVER_ROOT = "http://IP/seafhttp"
```

注意 `FILE_SERVER_ROOT` 是写在 python 配置文件中，所以需要加引号来表示这是个字符串。

如果还有问题，你可以使用 chrome/firefox 的调试模式来查看具体的错误信息。

网页上显示 "Page unavailable", 该怎么解决?

- 请检查 `logs/seahub_django_request.log` 来查看具体的错误信息。

安装完成后怎样修改 seafdata 的位置?

Seafdata 中使用 `ccnet/seafdata.ini` 来记录 seafdata 的位置

- 移动 seafdata 到起来位置
- 修改 `ccnet/seafdata.ini` 文件的内容

发送邮件失败，怎么排查?

请检查 `logs/seahub.log`，常见错误如下：

1. 检查配置文件 `seahub_settings.py` 中的项目是否正确。比如忘了加引号，`EMAIL_HOST_USER = XXX` 应该改成 `EMAIL_HOST_USER = 'XXX'`
2. 邮件服务器不可用。

Contribution

Licensing

Seafile and its desktop and mobile clients are published under the GPLv3.

The Seafile server's web end, i.e. Seahub, is published under the Apache License.

Discussion

Google Group: <https://groups.google.com/forum/?fromgroups#!forum/seafile>

IRC: #seafile on freenode

Follow us @seafile <https://twitter.com/seafile>

Reporting a Bug?

- Find the existing issues that fit your situation if any, or create a new issue. We are using github as our issue tracer <https://github.com/haiwen/seafile/issues?state=open>
- Join us on Google Group: <https://groups.google.com/forum/?fromgroups#!forum/seafile>

Code Style

The source code of seafile is ISO/IEC 9899:1999 (E) (a.k.a. C99) compatible. Take a look at [code standard](#).

Linux 下部署 Seafile 服务器

此文档用来说明如何使用预编译安装包来部署 Seafile 服务器。

使用安装脚本在 **Ubuntu 14.04** 或 **CentOS 7** 上快速安装

我们准备了一个安装脚本帮助您在 Ubuntu 14.04 或 CentOS 7 快速的安装部署 Seafile 服务(配置好 MariaDB, Memcached, WebDAV, Nginx 和开机自动启动脚本)：<https://github.com/haiwen/seafile-server-installer-cn>

在此基础上，您可以继续根据下面的文档来配置邮箱发送等服务。

家庭/个人 环境下部署 **Seafile** 服务器

- 部署 [Seafile 服务器（使用 SQLite）](#)

生产/企业 环境下部署 **Seafile** 服务器

企业环境下我们建议使用 MySQL 数据库，并将 Seafile 部署在 Nginx 或者 Apache 上，如果对于 Nginx 和 Apache 都不是很熟悉的话，我们建议使用 Nginx，相对于 Apache 来说，Nginx 使用起来比较简单。

基础功能:

- 部署 [Seafile 服务器（使用 MySQL）](#)
- [Nginx 下配置 Seahub](#)
- [Nginx 下启用 Hhttps](#)
- [Apache 下配置 Seahub](#)
- [Apache 下启用 Hhttps](#)
- [使用 Memcached](#)

高级功能:

- [开机启动 Seafile](#)
- [防火墙设置](#)
- [Logrotate 管理系统日志](#)

其他部署事项

Linux 下部署 Seafile 服务器

- [使用 NAT](#)
- [非根域名下部署 Seahub](#)
- [从 SQLite 迁移至 MySQL](#)

更多配置选项（比如开启用户注册功能），请查看 [服务器个性化配置](#)。

注意 如果在部署 Seafile 服务器时遇到困难

1. 阅读 [Seafile 组件](#) 以了解 Seafile 的运行原理。
2. [安装常见问题](#)。
3. 通过 Seafile 服务器管理员 QQ 交流群: 315229116 寻求帮助。

升级 Seafile 服务器

- [升级](#)

部署 Seafile 服务器（使用 SQLite）

本文档详细介绍如何使用预编译好的软件包来安装和运行 Seafile 服务器。

下载

到[下载页面](#)下载最新的服务器安装包。

部署和目录结构

注意: 如果你把 Seafile 文件放在一个外部存储的目录里（比如 NFS，CIFS），你应该使用 MySQL 而不是 SQLite 来作为数据库。请参考[下载和安装 Seafile 服务器（使用 MySQL）](#)。

假设你公司的名称为"haiwen",你也已经下载 seafile-server_1.4.0_* 到你的home 目录下。我们建议使用这样的目录结构:

```
mkdir haiwen
mv seafile-server_* haiwen
cd haiwen
#将 seafile-server_* 移动到 haiwen 目录下后
tar -xzf seafile-server_*
mkdir installed
mv seafile-server_* installed
```

现在，你的目录看起来应该像这样：

```
# tree . -L 2
.
├── installed
│   └── seafile-server_1.4.0_x86-64.tar.gz
└── seafile-server-1.4.0
    ├── reset-admin.sh
    ├── runtime
    ├── seafile
    ├── seafile.sh
    ├── seahub
    ├── seahub.sh
    ├── setup-seafile.sh
    └── upgrade
```

这样设计目录的好处在于

- 和 **seafile** 相关的配置文件都放在 **haiwen** 目录下，便于集中管理.
- 后续升级时,你只需要解压最新的安装包到 **haiwen** 目录下.

安装 **Seafile** 服务器

安装前的准备工作

安装 **Seafile** 服务器之前，请确认已安装以下软件

- python 2.7
- python-setuptools
- python-imaging
- python-ldap
- python-urllib3
- sqlite3

```
#Debian系统下
apt-get update
apt-get install python2.7 python-setuptools python-imaging pytho
n-ldap sqlite3 python-urllib3
```

```
# 在 CentOS 7 下
yum install python-setuptools python-imaging python-ldap MySQL-python python-memcached python-urllib3
```

安装

```
cd seafile-server-*
./setup-seafile.sh #运行安装脚本并回答预设问题
```

如果你的系统中没有安装上面的某个软件，那么 **Seafile** 初始化脚本会提醒你安装相应的软件包。该脚本会依次询问你一些问题，从而一步步引导你配置 **Seafile** 的各项参数。

| 参数 | 作用 | 说明 |
|-----------------------------|---|--|
| seafile server name | seafile 服务器的名字，目前该配置已经不再使用 | 3 ~ 15 个字符，可以用英文字母，数字，下划线 |
| seafile server ip or domain | seafile 服务器的 IP 地址或者域名 | 客户端将通过这个 IP 或者地址来访问你的 Seafile 服务 |
| seafile data dir | seafile 数据存放的目录，用上面的例子，默认将是 /data/haiwen/seafile-data | seafile 数据将随着使用而逐渐增加，请把它放在一个有足够大空闲空间的分区上 |
| seafile fileserver port | seafile fileserver 使用的 TCP 端口 | 一般使用默认的 8082 端口，如果已经被占用，可以设置为其他的端口 |

如果安装正确完成，会打印成功消息

现在你的目录结构将会是如下：

```
#tree haiwen -L 2
haiwen
├── conf                                # configuration files
│   ├── ccnet.conf
│   ├── seafile.conf
│   ├── seahub_settings.py
│   └── seafdav.conf
├── ccnet
│   ├── mykey.peer
│   ├── PeerMgr
│   └── seafile.ini
├── installed
│   └── seafile-server_1.4.0_x86-64.tar.gz
├── seafile-data
├── seafile-server-1.4.0               # active version
│   ├── reset-admin.sh
│   ├── runtime
│   ├── seafile
│   ├── seafile.sh
│   ├── seahub
│   ├── seahub.sh
│   ├── setup-seafile.sh
│   └── upgrade
├── seafile-server-latest              # symbolic link to seafile-server-1.4
.0
├── seahub-data
│   └── avatars
└── seahub.db
```

`seafile-server-latest` 文件夹是当前 **Seafile** 服务器文件夹的符号链接.将来你升级到新版本后, 升级脚本会自动更新使其始终指向最新的 **Seafile** 服务器文件夹.

启动运行 **Seafile** 服务器

启动 **Seafile** 服务器和 **Seahub** 网站

在 `seafile-server-1.4.0` 目录下, 运行如下命令:

- 启动 **Seafile**:


```
./seafile.sh start # 启动 Seafile 服务
```

- 启动 Seahub

```
./seahub.sh start <port> # 启动 Seahub 网站（默认运行在8000端口上）
```

小贴士: 你第一次启动 seahub 时, seahub.sh 脚本会提示你创建一个 seafile 管理员帐号。

服务启动后, 打开浏览器并输入以下地址

```
http://192.168.1.111:8000/
```

你会被重定向到登陆页面。输入你在之前创建的 Seafile 管理员帐号的用户名/密码即可。

恭喜! 现在你已经成功的安装了 Seafile 服务器。

在另一端口上运行 Seahub

如果你不想在默认的 8000 端口上运行 Seahub, 而是想自定义端口（比如8001）中运行, 请按以下步骤操作:

- 关闭 Seafile 服务器

```
./seahub.sh stop # 停止 Seafile 进程  
./seafile.sh stop # 停止 Seahub
```

- 更改 haiwen/conf/ccnet.conf 文件中 SERVICE_URL 的值(假设你的 ip 或者域名时 192.168.1.100), 如下 (从 5.0 版本开始, 可以直接在管理员界面中设置。注意, 如果同时在 Web 界面和配置文件中设置了这个值, 以 Web 界面的配置为准。):

```
SERVICE_URL = http://192.168.1.100:8001
```

- 重启 Seafile 服务器

```
./seafile.sh start # 启动 Seafile 服务
./seahub.sh start 8001 # 启动 Seahub 网站（运行在8001端口上）
```

关闭/重启 Seafile 和 Seahub

关闭

```
./seahub.sh stop # 停止 Seahub
./seafile.sh stop # 停止 Seafile 进程
```

重启

```
./seafile.sh restart # 停止当前的 Seafile 进程，然后重启 Seafile
./seahub.sh restart # 停止当前的 Seahub 进程，并在 8000 端口重新
启动 Seahub
```

如果停止/重启的脚本运行失败

大多数情况下 seafile.sh seahub.sh 脚本可以正常工作。如果遇到问题：

- 使用 `pgrep` 命令检查 seafile/seahub 进程是否还在运行中

```
pgrep -f seafile-controller # 查看 Seafile 进程
pgrep -f "seahub" # 查看 Seahub 进程
```

- 使用 `pkill` 命令杀掉相关进程

```
pkill -f seafile-controller # 结束 Seafile 进程
pkill -f "seahub" # 结束 Seafile 进程
```

部署 Seafile 服务器（使用 MySQL/MariaDB）

本文档用来说明通过预编译好的安装包来安装并运行基于 MySQL/MariaDB 的 Seafile 服务器。（MariaDB 是 MySQL 的分支）

下载

到[下载页面](#)下载最新的服务器安装包。

部署和目录设计

假设你公司的名称为 **haiwen**，你也已经下载 `seafile-server_1.4.0_*` 到你的 **home** 目录下。我们建议这样的目录结构：

```
mkdir haiwen
mv seafile-server_* haiwen
cd haiwen
#将 seafile-server_* 移动到 haiwen 目录下后
tar -xzf seafile-server_*
mkdir installed
mv seafile-server_* installed
```

现在，你的目录看起来应该像这样：

```
#tree haiwen -L 2
haiwen
├── installed
│   └── seafile-server_1.8.2_x86-64.tar.gz
└── seafile-server-1.8.2
    ├── reset-admin.sh
    ├── runtime
    ├── seafile
    ├── seafile.sh
    ├── seahub
    ├── seahub.sh
    ├── setup-seafile.sh
    └── upgrade
```

这样设计目录的好处在于

- 和 **seafile** 相关的配置文件都可以放在 **haiwen** 目录下，便于集中管理.
- 后续升级时,你只需要解压最新的安装包到 **haiwen** 目录下.

安装 Seafile 服务器

安装前的准备工作

安装 Seafile 服务器之前，请确认已安装以下软件

- MariaDB 或者 MySQL 服务器 (MariaDB 是 MySQL 的分支)
- python 2.7 (从 Seafile 5.1 开始，python 版本最低要求为2.7)
- python-setuptools
- python-imaging
- python-mysqldb
- python-ldap
- python-urllib3
- python-memcache (或者 python-memcached)

```
# 在Debian/Ubuntu系统下
apt-get update
apt-get install mariadb-server
apt-get install python2.7 python-setuptools python-imaging python-ldap python-mysqldb python-memcache python-urllib3
```

```
# 在 CentOS 7 下
yum install mariadb-server
yum install python-setuptools python-imaging python-ldap MySQL-python python-memcached python-urllib3
```

安装

```
cd seafile-server-*
./setup-seafile-mysql.sh #运行安装脚本并回答预设问题
```

如果你的系统中没有安装上面的某个软件，那么 **Seafile** 初始化脚本会提醒你安装相应的软件包。

该脚本会依次询问你一些问题，从而一步步引导你配置 **Seafile** 的各项参数：

| 参数 | 作用 | 说明 |
|-----------------------------|---|--|
| seafile server name | seafile 服务器的名字，目前该配置已经不再使用 | 3 ~ 15 个字符，可以用英文字母，数字，下划线 |
| seafile server ip or domain | seafile 服务器的 IP 地址或者域名 | 客户端将通过这个 IP 或者地址来访问你的 Seafile 服务 |
| seafile data dir | seafile 数据存放的目录，用上面的例子，默认将是 /data/haiwen/seafile-data | seafile 数据将随着使用而逐渐增加，请把它放在一个有足够大空闲空间的分区上 |
| seafile fileserver port | seafile fileserver 使用的 TCP 端口 | 该端口用于文件同步，请使用默认的 8082，不能更改。 |

在这里，你会被要求选择一种创建 **Seafile** 数据库的方式：

```
-----  
Please choose a way to initialize seafile databases:  
-----
```

- [1] Create new ccnet/seafile/seahub databases
- [2] Use existing ccnet/seafile/seahub databases

- 如果选择 1, 你需要提供根密码. 脚本程序会创建数据库和用户。
- 如果选择 2, ccnet/seafile/seahub 数据库应该已经被你 (或者其他人) 提前创建。

如果安装正确完成, 你会看到下面这样的输出 (新版本可能会有所不同)

```
-----  
Your seafile server configuration has been finished successfully.  
-----  
  
run seafile server:      ./seafile.sh { start | stop | restart }  
run seahub server:      ./seahub.sh { start <port> | stop | restart <port> }  
  
-----  
If you are behind a firewall, remember to allow input/output of these tcp ports:  
-----  
  
port of ccnet server:      10001  
port of seafile server:     12001  
port of httpserver server:  8082  
port of seahub:            8000  
  
When problems occur, Refer to  
  
    https://github.com/haiwen/seafile/wiki  
  
for information.
```

现在你的目录结构看起来应该是这样:

```
#tree haiwen -L 2
haiwen
├── conf                                # configuration files
│   ├── ccnet.conf
│   ├── seafile.conf
│   ├── seahub_settings.py
│   └── seafdav.conf
├── ccnet
│   ├── mykey.peer
│   ├── PeerMgr
│   └── seafile.ini
├── installed
│   └── seafile-server_1.8.2_x86-64.tar.gz
├── seafile-data
├── seafile-server-1.8.2               # active version
│   ├── reset-admin.sh
│   ├── runtime
│   ├── seafile
│   ├── seafile.sh
│   ├── seahub
│   ├── seahub.sh
│   ├── setup-seafile.sh
│   └── upgrade
├── seafile-server-latest               # symbolic link to seafile-server-1.8
.2
├── seahub-data
│   └── avatars
```

`seafile-server-latest` 文件夹为指向当前 **Seafile** 服务器文件夹的符号链接。将来你升级到新版本后，升级脚本会自动更新使其始终指向最新的 **Seafile** 服务器文件夹。

启动 **Seafile** 服务器

启动 **Seafile** 服务器和 **Seahub** 网站

在 `seafile-server-1.8.2` 目录下，运行如下命令

- 启动 **Seafile**:

```
./seafile.sh start # 启动 Seafile 服务
```

- 启动 Seahub

```
./seahub.sh start <port> # 启动 Seahub 网站（默认运行在8000端口上）
```

小贴士: 你第一次启动 seahub 时, seahub.sh 脚本会提示你创建一个 seafile 管理员帐号。

服务启动后, 打开浏览器并输入以下地址

```
http://192.168.1.111:8000/
```

你会被重定向到登陆页面. 输入管理员用户名和密码即可。

恭喜! 现在你已经成功的安装了 Seafile 服务器.

在另一端口上运行 Seahub

如果你不想在默认的 8000 端口上运行 Seahub, 而是想自定义端口（比如8001）中运行, 请按以下步骤操作:

- 关闭 Seafile 服务器

```
./seahub.sh stop # 停止 Seafile 进程  
./seafile.sh stop # 停止 Seahub
```

- 更改 haiwen/conf/ccnet.conf 文件中 SERVICE_URL 的值(假设你的 ip 或者域名时 192.168.1.100), 如下 (从 5.0 版本开始, 可以直接在管理员界面中设置。注意, 如果同时在 Web 界面和配置文件中设置了这个值, 以 Web 界面的配置为准。):

```
SERVICE_URL = http://192.168.1.100:8001
```

- 重启 Seafile 服务器

```
./seafile.sh start # 启动 Seafile 服务  
./seahub.sh start 8001 # 启动 Seahub 网站（运行在8001端口上）
```


关闭/重启 Seafile 和 Seahub

关闭

```
./seahub.sh stop # 停止 Seahub  
./seafile.sh stop # 停止 Seafile 进程
```

重启

```
./seafile.sh restart # 停止当前的 Seafile 进程，然后重启 Seafile  
./seahub.sh restart # 停止当前的 Seahub 进程，并在 8000 端口重新启动  
Seahub
```

如果停止/重启的脚本运行失败

大多数情况下 seafile.sh seahub.sh 脚本可以正常工作。如果遇到问题：

- 使用 **pgrep** 命令检查 seafile/seahub 进程是否还在运行中

```
pgrep -f seafile-controller # 查看 Seafile 进程  
pgrep -f "seahub" # 查看 Seahub 进程
```

- 使用 **pkill** 命令杀掉相关进程

```
pkill -f seafile-controller # 结束 Seafile 进程  
pkill -f "seahub" # 结束 Seahub 进程
```

OK!

查看seafile更多信息请访问:

- [Nginx 下配置 Seahub / Apache 下配置 Seahub](#)
- [Nginx 下启用 Hhttps / Apache 下启用 Hhttps](#)
- [Seafile LDAP配置](#)
- [管理员手册](#)

Nginx 下配置 Seahub

Nginx 环境下部署 Seahub/SeafServer

Seahub 是 Seafile 服务器的网站界面. SeafServer 用来处理浏览器端文件的上传与下载. 默认情况下, 它在 8082 端口上监听 HTTP 请求.

这里我们通过 fastcgi 部署 Seahub, 通过反向代理 (Reverse Proxy) 部署 SeafServer. 我们假设你已经将 Seahub 绑定了域名 "www.myseafile.com".

下面是一个 Nginx 配置文件的例子。

Ubuntu 下你可以

1. 创建文件 /etc/nginx/site-available/seafile.conf, 并拷贝以下内容
2. 删除 /etc/nginx/site-enabled/default: `rm /etc/nginx/site-enabled/default`
3. 创建符号链接: `ln -s /etc/nginx/sites-available/seafile.conf /etc/nginx/sites-enabled/seafile.conf`

```

server {
    listen 80;
    server_name www.myseafile.com;

    proxy_set_header X-Forwarded-For $remote_addr;

    location / {
        fastcgi_pass    127.0.0.1:8000;
        fastcgi_param    SCRIPT_FILENAME    $document_root$fastc
gi_script_name;
        fastcgi_param    PATH_INFO          $fastcgi_script_name
;

        fastcgi_param    SERVER_PROTOCOL    $server_protocol
;

        fastcgi_param    QUERY_STRING       $query_string;
        fastcgi_param    REQUEST_METHOD     $request_method;
        fastcgi_param    CONTENT_TYPE       $content_type;
        fastcgi_param    CONTENT_LENGTH     $content_length;
        fastcgi_param    SERVER_ADDR       $server_addr;
        fastcgi_param    SERVER_PORT       $server_port;
        fastcgi_param    SERVER_NAME       $server_name;
        fastcgi_param    REMOTE_ADDR       $remote_addr;

        access_log    /var/log/nginx/seahub.access.log;
        error_log    /var/log/nginx/seahub.error.log;
    }

    location /seafhttp {
        rewrite ^/seafhttp(.*)$ $1 break;
        proxy_pass http://127.0.0.1:8082;
        client_max_body_size 0;
        proxy_connect_timeout 36000s;
        proxy_read_timeout 36000s;
    }

    location /media {
        root /home/user/haiwen/seafhttp-server-latest/seahub;
    }
}

```

Nginx 默认设置 "client_max_body_size" 为 1M。如果上传文件大于这个值的话，会报错，相关 HTTP 状态码为 423 ("Request Entity Too Large")。你可以将值设为 0 以禁用此功能。

如果要上传大于 4GB 的文件，默认情况下 Nginx 会把整个文件存在一个临时文件中，然后发给上游服务器 (seaf-server)，这样容易出错。使用 1.8.0 以上版本同时在 Nginx 配置文件中设置以下内容能解决这个问题：

```
location /seafhttp {  
    ...  
    proxy_request_buffering off;  
}
```

修改 SERVICE_URL 和 FILE_SERVER_ROOT

下面还需要更新 SERVICE_URL 和 FILE_SERVER_ROOT 这两个配置项。否则无法通过 Web 正常的上传和下载文件。

5.0 版本开始，您可以通过管理员 Web 界面来设置这两个值 (注意，如果同时在 Web 界面和配置文件中设置了这个值，以 Web 界面的配置为准。):

```
SERVICE_URL: http://www.myseafile.com  
FILE_SERVER_ROOT: http://www.myseafile.com/seafhttp
```

5.0 版本之前需要修改 ccnet.conf 文件和 seahub_settings.py 文件

修改 ccnet.conf

```
SERVICE_URL = http://www.myseafile.com
```

修改 seahub_settings.py (增加一行，这是一个 python 文件，注意引号)

```
FILE_SERVER_ROOT = 'http://www.myseafile.com/seafhttp'
```

启动 **Seafile** 和 **Seahub**

```
./seafile.sh start  
./seahub.sh start-fastcgi
```

Nginx 下启用 Https

在 Seahub 端启用 https

免费 Self-Signed SSL 数字认证用户请看. 如果你是 SSL 付费认证用户可跳过此步.

通过 OpenSSL 生成 SSL 数字认证

```
openssl genrsa -out privkey.pem 2048
openssl req -new -x509 -key privkey.pem -out cacert.pem -days 1095
```

修改 Nginx 配置文件

请修改 nginx 配置文件以使用 HTTPS :

```
server {
    listen      80;
    server_name www.yourdoamin.com;
    rewrite ^ https://$http_host$request_uri? permanent;    #强制将
    http重定向到https
}

server {
    listen 443;
    ssl on;
    ssl_certificate /etc/ssl/cacert.pem;    #cacert.pem 文件路径
    ssl_certificate_key /etc/ssl/privkey.pem;    #privkey.pem 文件
    路径
    server_name www.yourdoamin.com;
    # .....
    fastcgi_param  HTTPS                      on;
    fastcgi_param  HTTP_SCHEME                 https;
}
```

配置文件示例

这里是配置文件示例:

```

server {
    listen      80;
    server_name www.yourdoamin.com;
    rewrite ^ https://$http_host$request_uri? permanent;    #
强制将http重定向到https
}
server {
    listen 443;
    ssl on;
    ssl_certificate /etc/ssl/cacert.pem;                      #cacert.pe
m 文件路径
    ssl_certificate_key /etc/ssl/privkey.pem;                #privkey.pem
文件路径
    server_name www.yourdoamin.com;
    proxy_set_header X-Forwarded-For $remote_addr;
    location / {
        fastcgi_pass      127.0.0.1:8000;
        fastcgi_param     SCRIPT_FILENAME    $document_root$fas
tcgi_script_name;
        fastcgi_param     PATH_INFO          $fastcgi_script_na
me;

        fastcgi_param     SERVER_PROTOCOL    $server_protocol;
        fastcgi_param     QUERY_STRING       $query_string;
        fastcgi_param     REQUEST_METHOD     $request_method;
        fastcgi_param     CONTENT_TYPE       $content_type;
        fastcgi_param     CONTENT_LENGTH     $content_length;
        fastcgi_param     SERVER_ADDR        $server_addr;
        fastcgi_param     SERVER_PORT        $server_port;
        fastcgi_param     SERVER_NAME        $server_name;
        fastcgi_param     REMOTE_ADDR        $remote_addr;
        fastcgi_param     HTTPS              on;
        fastcgi_param     HTTP_SCHEME        https;

        access_log        /var/log/nginx/seahub.access.log;
        error_log          /var/log/nginx/seahub.error.log;
    }
    location /seafhttp {
        rewrite ^/seafhttp(.*)$ $1 break;
        proxy_pass http://127.0.0.1:8082;
    }
}

```

```
        client_max_body_size 0;
        proxy_connect_timeout 36000s;
        proxy_read_timeout 36000s;
    }
    location /media {
        root /home/user/haiwen/seafile-server-latest/seahub;
    }
}
```

重新加载 Nginx

```
nginx -s reload
```

修改 **SERVICE_URL** 和 **FILE_SERVER_ROOT**

下面还需要更新 **SERVICE_URL** 和 **FILE_SERVER_ROOT** 这两个配置项。否则无法通过 Web 正常的上传和下载文件。

5.0 版本开始，您可以通过管理员 Web 界面来设置这两个值 (注意，如果同时在 Web 界面和配置文件中设置了这个值，以 Web 界面的配置为准。):

```
SERVICE_URL: https://www.myseafile.com
FILE_SERVER_ROOT: https://www.myseafile.com/seafhttp
```

启动 **Seafile** 和 **Seahub**

```
./seafile.sh start
./seahub.sh start-fastcgi
```


Apache 下配置 Seahub

请使用 Apache 2.4 版本。

准备工作

安装和启用需要的模块。

在 Ubuntu 系统上可以使用以下命令:

```
sudo a2enmod rewrite
sudo a2enmod proxy_fcgi
sudo a2enmod proxy_http
```

Apache 环境下部署 Seahub/FileServer

Seahub 是 Seafile 服务器的网站界面. FileServer 用来处理浏览器端文件的上传与下载. 默认情况下, 它在 8082 端口上监听 HTTP 请求.

这里我们通过 fastcgi 部署 Seahub, 通过反向代理 (Reverse Proxy) 部署 FileServer. 我们假设你已经将 Seahub 绑定了域名 "www.myseafile.com".

修改 Apache 配置文件: (`sites-enabled/000-default`) for ubuntu/debian
(`vhost.conf`) for centos/fedora

```

<VirtualHost *:80>
    ServerName www.myseafile.com
    # Use "DocumentRoot /var/www/html" for Centos/Fedora
    # Use "DocumentRoot /var/www" for Ubuntu/Debian
    DocumentRoot /var/www
    Alias /media /home/user/haiwen/seafhttp-seafile-server-latest/seahub
    /media

    RewriteEngine On

    <Location /media>
        Require all granted
    </Location>

    #
    # seafhttp fileserver
    #
    ProxyPass /seafhttp http://127.0.0.1:8082
    ProxyPassReverse /seafhttp http://127.0.0.1:8082
    RewriteRule ^/seafhttp - [QSA,L]

    #
    # seahub
    #
    SetEnvIf Request_URI . proxy-fcgi-pathinfo=unescape
    SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
    ProxyPass / fcgi://127.0.0.1:8000/
</VirtualHost>

```

修改 **SERVICE_URL** 和 **FILE_SERVER_ROOT**

下面还需要更新 **SERVICE_URL** 和 **FILE_SERVER_ROOT** 这两个配置项。否则无法通过 Web 正常的上传和下载文件。

5.0 版本开始，您可以通过管理员 Web 界面来设置这两个值(注意，如果同时在 Web 界面和配置文件中设置了这个值，以 Web 界面的配置为准。):

```

SERVICE_URL: http://www.myseafhttp.com
FILE_SERVER_ROOT: http://www.myseafhttp.com/seafhttp

```

5.0 版本之前需要修改 ccnet.conf 文件和 seahub_settings.py 文件

修改 **ccnet.conf**

```
SERVICE_URL = http://www.myseafile.com
```

修改 **seahub_settings.py**（增加一行，这是一个 **python** 文件，注意引号）

```
FILE_SERVER_ROOT = 'http://www.myseafile.com/seafhttp'
```

启动 **Seafile** 和 **Seahub**

```
sudo service Apache2 restart
./seafile.sh start
./seahub.sh start-fastcgi
```

其他说明

阅读[Seafile 组件](#)会帮你更好的理解 Seafile

在 Seafile 服务器端有两个组件：Seahub 和 FileServer。FileServer 通过监听 8082 端口处理文件的上传与下载。Seahub 通过监听 8000 端口负责其他的WEB页面。在 https 下, Seahub 应该通过 fastcgi 模式监听 8000 端口 (运行 ./seahub.sh start-fastcgi). 而且在 fastcgi 模式下, 如果直接访问 `http://domain:8000` 时, 会返回错误页面.

Apache 下启用 Https

请使用 Apache 2.4 版本。

通过 OpenSSL 生成 SSL 数字认证

免费 Self-Signed SSL 数字认证用户请看. 如果你是 SSL 付费认证用户可跳过此步.

```
openssl genrsa -out privkey.pem 2048
openssl req -new -x509 -key privkey.pem -out cacert.pem -days 10
95
```

在 Seahub 端启用 https

假设你已经按照[Apache 下配置 Seahub](#)对 Apache 进行了相关设置.请启用

```
mod_ssl
```

```
[sudo] a2enmod ssl
```

接下来修改你的 Apache 配置文件，这是示例:

```

<VirtualHost *:443>
    ServerName www.myseafile.com
    DocumentRoot /var/www

    SSLEngine On
    SSLCertificateFile /path/to/cacert.pem
    SSLCertificateKeyFile /path/to/privkey.pem

    Alias /media /home/user/haiwen/seafile-server-latest/seahub/m
edia

    <Location /media>
        ProxyPass !
        Require all granted
    </Location>

    RewriteEngine On

    #
    # seafile fileserver
    #
    ProxyPass /seafhttp http://127.0.0.1:8082
    ProxyPassReverse /seafhttp http://127.0.0.1:8082
    RewriteRule ^/seafhttp - [QSA,L]

    #
    # seahub
    #
    SetEnvIf Request_URI . proxy-fcgi-pathinfo=unescape
    SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
    ProxyPass / fcgi://127.0.0.1:8000/
</VirtualHost>

```

修改 **SERVICE_URL** 和 **FILE_SERVER_ROOT**

下面还需要更新 **SERVICE_URL** 和 **FILE_SERVER_ROOT** 这两个配置项。否则无法通过 Web 正常的上传和下载文件。

5.0 版本开始，您可以直接通过管理员 Web 界面来设置这两个值：(注意，如果同时在 Web 界面和配置文件中设置了这个值，以 Web 界面的配置为准。)

```
SERVICE_URL: https://www.myseafile.com  
FILE_SERVER_ROOT: https://www.myseafile.com/seafhttp
```

启动 **Seaf**file 和 **Seahub**

```
./seaf
```

file.sh start
./seahub.sh start-fastcgi

Seafile

升级指南

使用预编译 Seafile 服务器安装包的用户请看.

- 如果你是 [源码编译安装 Seafile](#) 的, 请参考那篇文档中的 升级 部分。
- 升级之后, 如果没有正常运行, 请清空 [Seahub 缓存](#)。

主版本升级 (比如从 **2.x** 升级到 **3.y**)

假设你现在使用 2.1.0 版本, 想要升级到 3.1.0 版本, 下载、解压新版本安装包后, 得到目录结构如下:

```
haiwen
-- seafile-server-2.1.0
-- seafile-server-3.1.0
-- ccnet
-- seafile-data
```

升级到 3.1.0:

1. 关闭 Seafile 服务 (如果正在运行):

```
cd haiwen/seafile-server-2.1.0
./seahub.sh stop
./seafile.sh stop
```

2. 查看 *seafile-server-3.1.0* 目录下的升级脚本:

```
cd haiwen/seafile-server-3.1.0
ls upgrade/upgrade_*
```

可以看到升级脚本文件如下:

```
...  
upgrade/upgrade_2.0_2.1.sh  
upgrade/upgrade_2.1_2.2.sh  
upgrade/upgrade_2.2_3.0.sh  
upgrade/upgrade_3.0_3.1.sh
```

3. 从当前版本（2.1.0）开始，按顺序运行以下脚本：

```
upgrade/upgrade_2.1_2.2.sh  
upgrade/upgrade_2.2_3.0.sh  
upgrade/upgrade_3.0_3.1.sh
```

4. 启动新版本 Seafile 服务器，完成升级：

```
cd haiwen/seafile-server-3.1.0/  
./seafile.sh start  
./seahub.sh start
```

小版本升级 (比如从 **3.0.x** 升级到 **3.2.y**)

假设你现在使用 3.0.0 版本，想要升级到 3.2.2 版本，下载、解压新版本安装包，得到目录结构如下：

```
haiwen  
-- seafile-server-3.0.0  
-- seafile-server-3.2.2  
-- ccnet  
-- seafile-data
```

升级到 3.2.2：

1. 关闭 Seafile 服务（如果正在运行）：

```
cd haiwen/seafile-server-3.0.0  
./seahub.sh stop  
./seafile.sh stop
```


2. 查看 `seafnle-server-3.2.2` 目录下的升级脚本：

```
cd haiwen/seafnle-server-3.2.2
ls upgrade/upgrade_*
```

可以看到升级脚本文件如下：

```
...
upgrade/upgrade_2.2_3.0.sh
upgrade/upgrade_3.0_3.1.sh
upgrade/upgrade_3.1_3.2.sh
```

3. 从当前版本（3.0.0）开始，按顺序运行以下脚本：

```
upgrade/upgrade_3.0_3.1.sh
upgrade/upgrade_3.1_3.2.sh
```

4. 启动新版本 **Seafnle** 服务器，完成升级：

```
cd haiwen/seafnle-server-3.2.2/
./seafnle.sh start
./seahub.sh start
```

维护版本升级 (比如从 3.1.0 升级到 3.1.2)

类似从 3.1.0 升级到 3.1.2，为维护版本升级。

1. 关闭 **Seafnle** 服务（如果正在运行）；
2. 对于此类升级，只需更新头像链接，直接运行升级脚本即可(因为历史原因，此升级脚本命名为 `minor-upgrade.sh`)：

```
cd seafnle-server-3.1.2
upgrade/minor-upgrade.sh
```

3. 运行升级脚本之后，启动新版本 **Seafnle** 服务器，完成升级；
4. 如果新版本运行正常，可以删除旧版本 **Seafnle** 文件。

```
rm -rf seafiler-server-3.1.0
```

开机启动 Seafile

Ubuntu 系统

使用 [/etc/init.d/](#) 来配置 Seafile/Seahub 开机启动.

创建 **/etc/init.d/seafile-server** 脚本

```
sudo vim /etc/init.d/seafile-server
```

脚本内容为: (同时需要修改相应的 `user` 和 `script_path` 字段的值)

```
#!/bin/bash
### BEGIN INIT INFO
# Provides:          seafile-server
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Seafile server
# Description:       Start Seafile server
### END INIT INFO

# 请将 user 改为你的Linux用户名
user=haiwen

# 请将 script_dir 改为你的 Seafile 文件安装路径
seafile_dir=/data/haiwen
script_path=${seafile_dir}/seafile-server-latest
seafile_init_log=${seafile_dir}/logs/seafile.init.log
seahub_init_log=${seafile_dir}/logs/seahub.init.log

# 若使用 Nginx/Apache, 请将其设置为true, 否者为 false
fastcgi=true
# fastcgi 端口, 默认为 8000.
fastcgi_port=8000

case "$1" in
```

```

        start)
            sudo -u ${user} ${script_path}/seafile.sh start
        >> ${seafile_init_log}
            if [ $fastcgi = true ];
            then
                sudo -u ${user} ${script_path}/seahub.sh
            start-fastcgi ${fastcgi_port} >> ${seahub_init_log}
            else
                sudo -u ${user} ${script_path}/seahub.sh
            start >> ${seahub_init_log}
            fi
        ;;
        restart)
            sudo -u ${user} ${script_path}/seafile.sh restar
        t >> ${seafile_init_log}
            if [ $fastcgi = true ];
            then
                sudo -u ${user} ${script_path}/seahub.sh
            restart-fastcgi ${fastcgi_port} >> ${seahub_init_log}
            else
                sudo -u ${user} ${script_path}/seahub.sh
            restart >> ${seahub_init_log}
            fi
        ;;
        stop)
            sudo -u ${user} ${script_path}/seafile.sh $1 >>
        ${seafile_init_log}
            sudo -u ${user} ${script_path}/seahub.sh $1 >> $
        {seahub_init_log}
        ;;
        *)
            echo "Usage: /etc/init.d/seafile-server {start|s
        top|restart}"
            exit 1
        ;;
    esac

```

注意: 如果使用本地 mysql 服务器, 请把 `# Required-Start: $remote_fs $syslog` 替换为 `# Required-Start: $remote_fs $syslog mysql`。

设置 **seafiler-sever** 脚本为可执行文件

```
sudo chmod +x /etc/init.d/seafiler-server
```

在 **rc.d** 中新增 **seafiler-server**

```
sudo update-rc.d seafiler-server defaults
```

完成

其他 **Debian** 系的 **Linux** 下

创建脚本 **/etc/init.d/seafiler-server**

```
sudo vim /etc/init.d/seafiler-server
```

脚本内容为: (同时需要修改相应的 `user` 和 `script_path` 字段的值)

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:          seafiler-server
# Required-Start:    $local_fs $remote_fs $network
# Required-Stop:     $local_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Starts Seafiler Server
# Description:       starts Seafiler Server
### END INIT INFO

# 请将 user 改为你的Linux用户名
user=haiwen

# 请将 script_path 改为你的 Seafiler 文件安装路径
seafiler_dir=/data/haiwen
script_path=${seafiler_dir}/seafiler-server-latest
seafiler_init_log=${seafiler_dir}/logs/seafiler.init.log
seahub_init_log=${seafiler_dir}/logs/seahub.init.log
```

```
# 若使用 fastcgi, 请将其设置为true
fastcgi=false
# fastcgi 端口, 默认为 8000.
fastcgi_port=8000

case "$1" in
    start)
        sudo -u ${user} ${script_path}/seafile.sh start
        >> ${seafile_init_log}
        if [ $fastcgi = true ];
        then
            sudo -u ${user} ${script_path}/seahub.sh
            start-fastcgi ${fastcgi_port} >> ${seahub_init_log}
        else
            sudo -u ${user} ${script_path}/seahub.sh
            start >> ${seahub_init_log}
        fi
        ;;
    restart)
        sudo -u ${user} ${script_path}/seafile.sh restart
        >> ${seafile_init_log}
        if [ $fastcgi = true ];
        then
            sudo -u ${user} ${script_path}/seahub.sh
            restart-fastcgi ${fastcgi_port} >> ${seahub_init_log}
        else
            sudo -u ${user} ${script_path}/seahub.sh
            restart >> ${seahub_init_log}
        fi
        ;;
    stop)
        sudo -u ${user} ${script_path}/seafile.sh $1 >>
        ${seafile_init_log}
        sudo -u ${user} ${script_path}/seahub.sh $1 >> $
        {seahub_init_log}
        ;;
    *)
        echo "Usage: /etc/init.d/seafile {start|stop|res
        tart}"

        exit 1

        ;;
```

```
esac
```

注意:

- 如果你想在 `fastcgi` 下运行 `Seahub`,请设置 `fastcgi` 变量为 `true`
- 如果使用本地 `mysql` 服务器,请把 `# Required-Start: $remote_fs $syslog` 替换为 `# Required-Start: $remote_fs $syslog mysql`。

为日志文件创建目录

```
mkdir /path/to/seafile/dir/logs
```

设置 **seafile-sever** 脚本为可执行文件

```
sudo chmod +x /etc/init.d/seafile-server
```

在 **rc.d** 中新增 **seafile-server**

```
sudo update-rc.d seafile-server defaults
```

完成

RHEL/CentOS 系统统方法 1

RHEL/CentOS 下,[/etc/rc.local](#) 脚本会随系统开机自动执行,所以我们在这个脚本中设置启动 Seafile/Seahub.

- 定位 python(python 2.6 or 2.7)

```
which python2.6 # or "which python2.7"
```

- 在 `/etc/rc.local` 脚本中,将 `python2.6(2.7)`路径加入到 **PATH** 字段中,并增加 `Seafile/Seahub` 启动命令

```
、  
  
# 假设 python 2.6(2.7) 可执行文件在 /usr/local/bin 目录下  
PATH=$PATH:/usr/local/bin/  
  
# 请将 user 改为你的Linux用户名  
user=haiwen  
  
# 请将 script_path 改为你的 Seafile 文件安装路径  
seafile_dir=/data/haiwen  
script_path=${seafile_dir}/seafile-server-latest  
  
sudo -u ${user} ${script_path}/seafile.sh start > /tmp/seafile.i  
nit.log 2>&1  
sudo -u ${user} ${script_path}/seahub.sh start > /tmp/seahub.ini  
t.log 2>&1
```

注意: 如果你想在fastcgi下启动Seahub,只需将上文中最后一行"**seahub.sh start**"改为"**seahub.sh start-fastcgi**"

RHEL/CentOS 系统方法 2

RHEL/CentOS 下,我们通过 /etc/init.d/ 脚本将 Seafile/Seahub作为服务程序随开机启动.

创建**/etc/sysconfig/seafile**文件


```
# 请将 user 改为你的Linux用户名
user=haiwen

# 请将 script_path 改为你的 Seafile 文件安装路径
seafile_dir=/home/haiwen
script_path=${seafile_dir}/seafile-server-latest
seafile_init_log=${seafile_dir}/logs/seafile.init.log
seahub_init_log=${seafile_dir}/logs/seahub.init.log

# 若使用 fastcgi, 请将其设置true
fastcgi=false

# fastcgi 端口, 默认为 8000.
fastcgi_port=8000
```

创建/etc/init.d/seafile文件

```
#!/bin/bash
#
# seafile

#
# chkconfig: - 68 32
# description: seafile

# Source function library.
. /etc/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

if [ -f /etc/sysconfig/seafile ];then
    . /etc/sysconfig/seafile
else
    echo "Config file /etc/sysconfig/seafile not found!
Bye."
    exit 200
fi

RETVAL=0
```

```

start() {
    # Start daemons.
    echo -n "Starting seafile: "
    ulimit -n 30000
    su - ${user} -c"${script_path}/seafile.sh start >> ${seaf
file_init_log} 2>&1"
    RETVAL=$?
    echo
    [ $RETVAL -eq 0 ] && touch /var/lock/subsys/seafile
    return $RETVAL
}

stop() {
    echo -n "Shutting down seafile: "
    su - ${user} -c"${script_path}/seafile.sh stop >> ${seaf
ile_init_log} 2>&1"
    RETVAL=$?
    echo
    [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/seafile
    return $RETVAL
}

# See how we were called.
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart|reload)
        stop
        start
        RETVAL=$?
        ;;
    *)
        echo $"Usage: $0 {start|stop|restart}"
        RETVAL=3
esac

exit $RETVAL

```

创建/etc/init.d/seahub脚本

```
#!/bin/bash
#
# seahub

#
# chkconfig: - 69 31
# description: seahub

# Source function library.
. /etc/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

if [ -f /etc/sysconfig/seafile ];then
    . /etc/sysconfig/seafile
else
    echo "Config file /etc/sysconfig/seafile not found!
Bye."
    exit 200
fi

RETVAL=0

start() {
    # Start daemons.
    echo -n "Starting seahub: "
    ulimit -n 30000
    if [ $fastcgi = true ];
    then
        su - ${user} -c"${script_path}/seahub.sh start-f
astcgi ${fastcgi_port} >> ${seahub_init_log} 2>&1"
    else
        su - ${user} -c"${script_path}/seahub.sh start >
> ${seahub_init_log} 2>&1"
    fi
    RETVAL=$?
    echo
    [ $RETVAL -eq 0 ] && touch /var/lock/subsys/seahub
    return $RETVAL
}
```

```

}

stop() {
    echo -n "Shutting down seafile: "
    su - ${user} -c"${script_path}/seahub.sh stop >> ${seahu
b_init_log} 2>&1"
    RETVAL=$?
    echo
    [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/seahub
    return $RETVAL
}

# See how we were called.
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart|reload)
        stop
        start
        RETVAL=$?
        ;;
    *)
        echo $"Usage: $0 {start|stop|restart}"
        RETVAL=3
esac

exit $RETVAL

```

接下来启动服务程序:

```

chmod 550 /etc/init.d/seafile
chmod 550 /etc/init.d/seahub
chkconfig --add seafile
chkconfig --add seahub
chkconfig seahub on
chkconfig seafile on

```

开机启动 Seafile

执行:

```
service seafile start  
service seahub start
```

完成

防火墙设置

默认情况下，Seafile 开启了以下两个端口：

| | |
|------------|------|
| Seahub | 8000 |
| FileServer | 8082 |

如果你在 Nginx/Apache 下运行 Seafile，并且使用了 HTTPS, 开启 443 端口即可。

在服务器端设置 logrotate

工作原理

自 3.1 版本以后，seaf-server 和 ccnet-server 支持通过接收 SIGUSR1 信号来管理日志文件。

这个功能在你需要剪切日志文件但是不想关闭服务器的时候非常有用。

注意: 此功能在 Windows 下并不适用

logrotate 默认配置

对于 Debian, logrotate 默认存储在 `/etc/logrotate.d/`

配置示例

假设你的 ccnet-server 的日志文件是 `/home/haiwen/logs/ccnet.log` , ccnet-server 进程的 pidfile 是 `/home/haiwen/pids/ccnet.pid` . seaf-server's 的日志文件是 `/home/haiwen/logs/seaf-server.log` , seaf-server 进程的 pidfile 是 `/home/haiwen/pids/seaf-server.pid` :

则请按如下配置 logrotate:

```
/home/haiwen/logs/seaf-server.log
{
    daily
    missingok
    rotate 52
    compress
    delaycompress
    notifempty
    sharedscripts
    postrotate
        [ ! -f /home/haiwen/pids/seaf-server.pid ] || kill
11 -USR1 `cat /home/haiwen/pids/seaf-server.pid`
    endscript
}

/home/haiwen/logs/ccnet.log
{
    daily
    missingok
    rotate 52
    compress
    delaycompress
    notifempty
    sharedscripts
    postrotate
        [ ! -f /home/haiwen/pids/ccnet.pid ] || kill -US
R1 `cat /home/haiwen/pids/ccnet.pid`
    endscript
}
```

对于 Debian 用户, 可以将以上配置文件存储在 `/etc/logrotate.d/seaf`

使用 memcached

安装 Memcached 能够显著提高系统性能。

首先你需要保证 `libmemcached` 库已经安装在你的系统中。要想使用 memcached 集群，我们要求使用 1.0.18 或者更新的版本。

使用单节点 memcached

在 CentOS 7 上

```
sudo yum install gcc libffi-devel python-devel openssl-devel libmemcached libmemcached-devel
sudo pip install pylibmc
sudo pip install django-pylibmc
```

将以下配置添加到 `seahub_settings.py` 中：

```
CACHES = {
    'default': {
        'BACKEND': 'django_pylibmc.memcached.PyLibMCCache',
        'LOCATION': '127.0.0.1:11211',
    }
}
```

最后重启 Seahub 以使更改生效：

```
./seahub.sh restart
```

如果更改没有生效，请删除 `seahub_setting.pyc` 缓存文件。

使用 memcached 集群

在 Ubuntu 16.04 或者更新的系统上，可以直接使用 `apt-get` 来安装。

```
sudo apt-get install libmemcached-dev
```

使用 Memcached

在其他比较老的系统，比如 CentOS 7 或者 Ubuntu 14.04 上，你需要从源代码编译安装这个库。

```
sudo apt-get install build-essential # or sudo yum install gcc g
cc-c++ make openssl-devel libffi-devel python-devel
wget https://launchpad.net/libmemcached/1.0/1.0.18/+download/lib
memcached-1.0.18.tar.gz
tar zxf libmemcached
cd libmemcached-1.0.18
./configure
make
sudo make install
```

然后安装相关的 Python 库。

```
sudo pip install pylibmc
sudo pip install django-pylibmc
```

使用的是 memcached 集群（即同时使用多台 memcached 服务器），`CACHES` 变量应该设置如下。这个配置使用一致性哈希的方式把 key 分布在多个 memcached 服务器上。更多的信息可以参考 [pylibmc 文档](#) 和 [django-pylibmc 文档](#)

```
CACHES = {
    'default': {
        'BACKEND': 'django_pylibmc.memcached.PyLibMCCache',
        'LOCATION': ['192.168.1.134:11211', '192.168.1.135:11211',
                    '192.168.1.136:11211'],
        'OPTIONS': {
            'ketama': True,
            'remove_failed': 1,
            'retry_timeout': 3600,
            'dead_timeout': 3600
        }
    }
}
```

最后重启 Seahub 以使更改生效：

```
./seahub.sh restart
```

使用 Memcached

如果更改没有生效，请删除 `seahub_setting.pyc` 缓存文件.

防火墙 / NAT 设置

通过广域网(WAN)访问部署在局域网(LAN)的 Seafile 服务器,需要:

- 一台支持端口转发的路由器
- 使用动态域名解析服务
- 配置 Seafile 服务器

在路由器中设置端口转发

确保路由器支持端口转发功能

首先, 确保你的路由器支持端口转发功能:

- 根据路由器管理手册操作说明(或网络搜索), 进入路由器的管理用户界面。
- 找到包含 "转发" 或者 "高级" 等关键词的页面, 说明此路由器支持端口转发功能。

设置路由转发规则

Seafile 服务器包含两个组件, 请根据以下规则为 Seafile 组件设置端口转发。

| 组件 | 默认端口 |
|------------|------|
| fileserver | 8082 |
| seahub | 8000 |

- 如果是在 Apache/Nginx 环境下部署的 Seafile, 则不需要打开 8000 和 8082 端口, 只需要 80 或 443 端口即可。
- 以上是默认端口设置, 具体配置可自行更改。

端口转发测试

设置端口转发后, 可按以下步骤测试是否成功:

- 打开一个命令行终端
- 访问 `http://who.is` 得到本机的IP
- 通过以下命令连接 Seahub

```
telnet <Your WAN IP> 8000
```

如果端口转发配置成功，命令行会提示连接成功。否则，会显示 *connection refused* 或者 *connection timeout*，提示连接不成功。

若未成功，原因可能如下：

- 端口转发配置错误
- 需要重启路由器
- 网络不可用

设置 **SERVICE_URL** 和 **FILE_SERVER_ROOT**

服务器依赖于 `ccnet.conf` 中的 "SERVICE_URL" 和 `seahub_setting.py` 中的 `FILE_SERVER_ROOT` 来生成文件的上传/下载链接 (从 5.0 开始这两个值可以通过 Web 界面来设置)。如果使用内置的 web 服务器，改为

```
SERVICE_URL = http://<Your WAN IP>:8000
```

如果配置了 Nginx，则需要修改为

```
SERVICE_URL = http://<Your WAN IP>  
FILE_SERVER_ROOT = http://<Your WAN IP>/seafhttp
```

大部分路由器都支持 NAT loopback。当你通过内网访问 Seafile 时，即使使用外部 IP，流量仍然会直接通过内网走。

使用域名解析服务

为什么使用动态域名解析服务？

完成以上端口转发配置工作后，就可以通过外网 IP 访问部署在局域网内的 Seafile 服务器了。但是对于大多数人来说，外网 IP 会被 ISP (互联网服务提供商) 定期更改，这就使得，需要不断的进行重新配置。

使用 NAT

可以使用动态域名解析服务来解决这个问题。通过使用域名解析服务，你可以通过域名（而不是 IP）来访问 Seahub，即使 IP 会不断变化，但是域名始终会指向当前 IP。

互联网上提供域名解析服务的有很多，我们推荐 www.noip.com。

怎样使用域名解析服务，不在本手册说明范围之内，但是基本上，你需要遵循以下步骤：

1. 选择一个域名解析服务提供商。
2. 注册成为此服务商的一个用户。

更改 Seafile 配置

当你配置好域名解析服务之后，需要对 `ccnet.conf` 进行更改 (或者通过管理员 Web 界面来修改)：

```
SERVICE_URL = http://<你的域名>:8000
```

然后重新 Seafile 服务。

网络设置

你如果使用内置的服务器，需要开启 8000 和 8082 两个端口。如果你的 Seafile 服务器是运行在 Nginx/Apache 环境下，并且开启了 HTTPS，则需要开启 443 端口。

从 SQLite 迁移到 MySQL

首先请确认 MySQL 的 Python 模块已经安装. Ubuntu 下, 安装命令为 `apt-get install python-mysqldb` .

请按以下步骤操作:

1. 停止 Seafile 和 Seahub
2. 下载 [sqlite2mysql.sh](#) 和 [sqlite2mysql.py](#) 到 Seafile 的安装根目录 (`/data/haiwen`) 里.
3. 运行 `sqlite2mysql.sh` 脚本

```
chmod +x sqlite2mysql.sh
./sqlite2mysql.sh
```

这个脚本将生成三个文件: `ccnet-db.sql` , `seafile-db.sql` , `seahub-db.sql` .

4. 新建3个数据库, 分别命名为 `ccnet-db` , `seafile-db` , `seahub-db` .

```
create database `ccnet-db` character set = 'utf8';
create database `seafile-db` character set = 'utf8';
create database `seahub-db` character set = 'utf8';
```

5. 修改 `/etc/my.conf`, 添加下列语句, 并重启 mysql (`sudo service mysql restart`), 这个语句主要是确保数据库使用 UTF8 编码

```
[mysqld]
collation-server = utf8_unicode_ci
init-connect='SET NAMES utf8'
character-set-server = utf8
```

6. 运行 sql 文件:

```
mysql> use `ccnet-db`  
mysql> source ccnet-db.sql  
mysql> use `seafile-db`  
mysql> source seafile-db.sql  
mysql> use `seahub-db`  
mysql> source seahub-db.sql
```

7. 更改配置

在 `conf/ccnet.conf` 中增加以下语句:

```
[Database]  
ENGINE=mysql  
HOST=127.0.0.1  
USER=root  
PASSWD=root  
DB=ccnet-db  
CONNECTION_CHARSET=utf8
```

注意: 使用 `127.0.0.1`, 不要使用 `localhost` .

将 `seafile.conf` 中的数据库配置信息更改为以下语句:

```
[database]  
type=mysql  
host=127.0.0.1  
user=root  
password=root  
db_name=seafile-db  
CONNECTION_CHARSET=utf8
```

在 `seahub_settings.py` 中增加以下语句:


```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'USER' : 'root',  
        'PASSWORD' : 'root',  
        'NAME' : 'seahub-db',  
        'HOST' : '127.0.0.1',  
        'OPTIONS': {  
            "init_command": "SET storage_engine=INNODB",  
        }  
    }  
}
```

8. 重启 Seafile and Seahub

5.0.0 版本中的配置文件位置改动

Seafile 服务器由几个组件组成，每个组件都有自己的配置文件。5.0 版本之前这些文件放在不同的目录下，管理起来不太方便。

5.0.0 之前的各个配置文件分布如下：

```
└─ seahub_settings.py
└─ ccnet/
    └─ ccnet.conf
└─ seafile/
    └─ seafile.conf
└─ conf/
    └─ seafdav.conf
└─ pro-data/
    └─ seafevents.conf # (专业版)
└─ seafile-server-latest/
```

5.0.0 之后，所有的配置文件都集中放置在 **conf** 目录下：

```
└─ conf/
    └─ ccnet.conf
    └─ seafile.conf
    └─ seafdav.conf
    └─ seahub_settings.py
    └─ seafevents.conf # (专业版)
└─ ccnet/
└─ seafile/
└─ pro-data/
```

这样把所有的配置文件都集中放置，管理起来就更方便了。

当您升级到 Seafile 5.0.0 版本时，升级脚本会自动帮您把上述文件移到 **conf** 目录下。

安装与升级

我们测试用的系统是 Windows 2008 server R2 SP1。

- [下载安装 Windows 版 Seafile 服务器](#)
- [安装 Seafile 为 Windows 服务](#)
- [所用端口说明](#)
- [升级](#)

注意：默认情况下，Seafile 需要用到 8000, 8082 两个端口。

服务器管理

- [垃圾回收不再需要的数据块](#)

常见问题

如果您安装 Seafile 服务器失败，请首先查看 `seafserv-applet.log` 文件。

安装完后，本地网页无法打开

确保您使用的是 Python 2.7.11 32 位版本。

"ERROR: D:/seafile-server\seahub.db not found"

此文件是在 Seafile 初始化过程中创建的。请执行下面两步：

1. 检查您的 Python 以及 Python 环境变量是否设置正确。
2. 将您的 Seafile 服务器包放在一个简短的路径下，比如 `C:\seafile-packages`。

创建 `seahub.db` 文件失败

请使用 Python 2.7.4 32 位版本，不要使用 Python 3.0 及以上版本。

不能通过 **Web** 端上传或下载文件

请先确保您已经正确设置了 `SERVICE_URL` 和 `FILE_SERVER_ROOT` 。这可以通过 Web 端"管理员界面->设置"中更改。

浏览器不能获得 **css** 和 **javascript** 文件

1. 使用 python 2.7.11 32 位版本。如果您已经安装了 python 的其他版本，请先卸载然后安装 python 2.7.11 版本。重启 Seafile 服务器确认此问题是否依然存在。
2. 将注册表路径 `HKEY_CLASSES_ROOT\MIME\Database\Content Type` 下的非 ASCII 键删除，然后重试。

如何移动 **seafile-server** 文件夹

假设你的 Seafile 服务器程序位置为 `C:/SeafileProgram`，数据文件夹位置为 `D:/seafile-server`。现在你希望把数据文件夹从 `D:/seafile-server` 移动到 `E:/seafile-server`

1. 先在托盘菜单里选 "停止并退出 **seafile** 服务器"
2. 把数据文件夹 `D:/seafile-server` 移动到新位置 `E:/seafile-server`
3. 打开 `C:/SeafileProgram` 文件夹下的 `seafserv.ini` 这个文件。这个文件记录了数据文件夹的路径。把这个文件的内容改为 `E:/seafile-server`。注意：如果你的新位置的路径包含非英文字符，那么请用支持 UTF8 格式的文本编辑器来编辑 `seafserv.ini` 文件，并保存为 UTF8 格式。否则 Seafile 服务器程序可能无法正确读取这个文件的内容。
4. 重新启动 Seafile 服务器。

更多信息

想了解更多关于 Seafile 服务器的信息，请访问

<https://github.com/haiwen/seafile/wiki>

下载安装 Windows 版 Seafile 服务器

安装 Python 2.7.11 32 位版本

- 下载并安装 [python 2.7.11 32 位版本](#)
- 将 python2.7 的安装路径添加到系统的环境变量中 (PATH 变量)。比如：如果您将 python 2.7.11 安装在 C:\Python27 路径下，那么就将 C:\Python27 添加到环境变量中。

注意：一定要使用 Python 2.7.11 32 位版本。64 位版本或不是 2.7.11 的版本不能工作。

下载并解压 Seafile 服务器

- 获取 [Seafile 服务器](#) 的最新版本。
- 为 Seafile 服务器程序创建一个新的文件夹，比如 C:\SeafileProgram\。请记住此文件夹的位置，我们将在以后用到它。
- 将 **seafile-server_5.0.3_win32.tar.gz** 解压到 C:\SeafileProgram\ 目录下。

现在，您的目录结构应该像如下这样：

```
C:\SeafileProgram
    |__ seafile-server-5.0.3
```

启动与初始化

启动 Seafile 服务器

在 C:\SeafileProgram\seafile-server-5.0.3\ 文件夹下，找到 **run.bat** 文件并双击，以启动 Seafile 服务器。此时，您应该注意到 Seafile 服务器的图标已经出现在您的系统托盘中。

选择一个磁盘作为 Seafile 服务器数据的存储位置

现在，您可以在弹出的对话框中选择一个磁盘，以便存储 Seafile 服务器的数据：

- 请确保选择的磁盘拥有足够的剩余空间
- 点击确认按钮后，Seafile 将会在您选择的磁盘下为您创建一个名为 `seafile-server` 的文件夹。这个文件夹就是 Seafile 服务器的数据文件夹。如果您选择 D 盘，那么数据文件夹为 `D:\seafile-server`

添加管理员帐号

右击 Seafile 服务器的系统托盘图标，选择"添加管理员帐号"选项。在弹出的对话框中输入您的管理员用户名和密码。

如果操作成功，Seafile 服务器托盘图标处会弹出一个气泡提示您"添加 Seahub 管理员账户成功"

配置 Seafile 服务器

初始化服务器之后，还需配置以下选项，否则不能进行文件的上传下载：

- 访问服务器的 Web 界面 (打开 `http://<您的 IP 地址>:8000`)，用管理员账号登录
- 点击左上角的扳手图标，进入管理员界面，在进入"设置"标签
- 将 **SERVICE_URL** 的值配置成 `http://<您的 IP 地址>:8000`。比如您的 Windows 服务器地址为 `192.168.1.100`，那么配置成 `SERVICE_URL = http://192.168.1.100:8000`
- 将 **FILE_SERVER_ROOT** 的值配置成 `http://<您的 IP 地址>:8082`。比如您的 Windows 服务器地址为 `192.168.1.100`，那么配置成 `SERVICE_URL = http://192.168.1.100:8082`

配置完成

Seafile 服务器的配置到此已经完成。如果您想了解如何使用 Seafile 客户端，请参考 [Seafile 客户端手册](#)

您可能还会想要了解以下信息：

- [Seafile LDAP 配置](#)
- [安装 Seafile 为 Windows 服务](#)
- [所用端口说明](#)
- [升级](#)
- [个性化配置](#)

安装 Seafile 为 Windows 服务

将 Seafile 服务器作为 Windows 服务安装的好处

- 在您的所有用户注销后 Seafile 服务器能够继续保持运行
- 系统启动时，即使没有用户登录，Seafile 服务器也会开始运行

如何作为 Windows 服务安装

- 右击 Seafile 服务器托盘图标，选择"安装为 Windows 服务"选项
- 在弹出的对话框中，点击是按钮

如果操作成功，将会弹出一个对话框提示您"已经成功安装 Seafile 服务"。

确认 Seafile 服务器已经开始作为 Windows 服务运行

- 注销当前用户
- 在另一台电脑上访问 Seahub。如果 Seahub 网站仍然可以访问，那么说明 Seafile 服务器已经开始作为 Windows 服务运行

安装为 Windows 服务后如何启动托盘图标

如果您已经将 Seafile 服务器安装为 Windows 服务，那么在您下次系统启动时，Seafile 服务将会在后台自动运行。这样，当用户登录时，Seafile 服务器托盘图标就不会自动出现。

启动托盘图标，只需双击 `C:\SeafileProgram\seafile-server-1.7.0` 文件夹下的 `run.bat` 文件。

卸载 Seafile 服务器的 Windows 服务

如果您想卸载 Seafile 服务器的 Windows 服务，请执行以下两步：

- 右击托盘图标，选择"卸载 Windows 服务"选项
- 在弹出的确认对话框中点击"是"按钮

所用端口说明

Seafile 服务器由两个组件组成，默认情况下用到 8000, 8082 两个端口号 (TCP)。

配置文件

所有端口的相关配置都记录在 `ccnet.conf` 文件和 `seafile.conf` 文件中

如何打开 `ccnet.conf` 文件

- 右击 Seafile 服务器托盘图标，选择"打开 `seafile-server` 文件夹"选项
- 打开 `seafile-server` 目录下的 `conf` 文件夹。`ccnet.conf` 文件就在此文件夹下。

如何打开 `seafile.conf` 文件

- 右击 Seafile 服务器托盘图标，选择"打开 `seafile-server` 文件夹"选项
- 打开 `seafile-server` 目录下的 `conf` 文件夹。`seafile.conf` 文件就在此文件夹下。

在接下来的部分，我们分别列举了 Seafile 服务器各个组件用到的TCP端口以及如何改变它们（比如，一些端口很有可能已经被其他应用程序占用）。

注意：如果您改变了以下任何端口，请重启 Seafile 服务器。

seahub

`seahub` 是 Seafile 服务器的 Web 端。

注意：如果您改变了 Seahub 的端口号，"管理员界面->设置"中的 `SERVICE_URL` 也需要做相应的改动。

- 默认端口：8000
- 如何设置端口号：编辑 `seafile.conf` 文件。设置在 `seahub` 段下 `port` 的值。

```
[seahub]
port=8000
```

- "管理员界面->设置" 中的 `SERVICE_URL`。比如，如果您将端口号重新设置为 8001，那么更改 `SERVICE_URL` 的值如下：

```
SERVICE_URL = <您的 IP 或者域名>:8001
```

seafile fileserver

`seafile fileserver` 负责为 Seahub 处理文件的上传和下载

- 默认端口：8082
- 如何设置端口号：桌面客户端会连接这个端口来同步文件，所以不要修改这个端口。

升级

- [小版本升级](#)
- [大版本升级](#)
- [升级 Windows 服务](#)

注意：升级之前，你需要先停止 Seafile 服务器

解压新版本服务器

假设升级之前，你的目录结构是：

```
C:/SeafileProgram
    |_____ seafile-server-1.7.0/
```

那么，升级的第一步是下载新版本的程序包，并解压到文件夹 ‘C:/SeafileProgram` 下面。

```
C:/SeafileProgram
    |_____ seafile-server-1.7.0/
    |_____ seafile-server-1.8.0/
```

小版本升级 (如从 **1.7.0** 版本升级到 **1.7.1** 版本)

现在假定您要将 Seafile 服务器的 Windows 服务从 1.7.0 版本升级到 1.7.1 版本

迁移 **avatars** 文件夹的内容

找到 **seafile-server-1.7.0/seahub/media/avatars** 目录

在 **avatars**/ 文件夹中包含着所有 Seafile 用户的头像。

如果您有一个用户名为 `foo@foo.com` 的用户，那么在 **avatars**/ 文件夹中，您会发现一个叫作 `foo@foo.com` 的子文件夹。这个子文件夹包含着用户 `foo@foo.com` 的头像图片。

升级

将所有像 `foo@foo.com` 的这种子文件夹拷贝到 `seafile-server-1.7.1/seahub/media/avatars` 目录下。这样，当您启动新的 1.7.1 版本的 Seafile 服务器时，这些头像可以正确加载。

大版本升级 (如从 1.7.0 版本升级到 1.8.0 版本)

现在假定您要将 Seafile 服务器的 Windows 服务从 1.7.x 版本升级到 1.8.y 版本

运行数据库升级脚本

- 找到 `seafile-server-1.8.y/upgrade` 目录
- 在这个目录下，右击 `upgrade_1.7_1.8.bat` 文件
- 选择"以管理员身份运行"

拷贝头像

将在 `seafile-server-1.7.0/seahub/media/avatars` 目录下的所有子文件夹拷贝到 `seafile-server-1.8.0/seahub/media/avatars` 目录下

升级 Windows 服务

如果您已经将 Seafile 服务器作为 Windows 服务安装，您需要做以下几步：

- 运行老版本的 Seafile Windows 服务器，右击托盘图标，在菜单中选择卸载 **Windows 服务**
- 退出老版本的 Seafile Windows 服务器
- 启动新版本的 Seafile Windows 服务器，右击托盘图标，在菜单中选择安装为 **Windows 服务**

服务器从 Windows 迁移到 Linux

假设你当前已经在使用 Windows 服务器(使用 SQLite 数据库)，现在希望把服务器迁移到 Linux 下。

1. 安装 Linux 服务器

第一步你需要安装全新一个 Linux 服务器。同样使用 SQLite 数据库。下面假设你把 Seafile 服务器默认安装在 `/home/haiwen/` 目录下。

2. 替换数据和配置文件

删除 Linux 的配置文件和数据

```
rm /home/haiwen/seahub_settings.py
rm /home/haiwen/seahub.db
rm -r /home/haiwen/seafile-data
cp /home/haiwen/ccnet/seafile.ini /home/haiwen/seafile.ini
rm -r /home/haiwen/ccnet
```

其中 `seafile.ini` 指向 `seafile-data` 目录所在位置，等会需要用到，这里先拷贝出来。

拷贝配置文件和数据

- 将 Windows 中 **seafile-server** 文件夹下的 `seahub_settings.py` 文件，拷贝到 linux `/home/haiwen/` 目录下
- 将 Windows 中 **seafile-server** 文件夹下的 `seahub.db` 文件，拷贝到 linux `/home/haiwen/` 目录下；
- 将 Windows 中 **seafile-server** 的子文件夹 `seafile-data`，拷贝到 linux `/home/haiwen/` 目录下；
- 将 Windows 中 **seafile-server** 的子文件夹 `ccnet`，拷贝到 linux `/home/haiwen/` 目录下；
- 将 `/home/haiwen/seafile.ini` 拷贝到新 **ccnet** 目录中

垃圾回收

- 右击 Seafile 托盘图标，选择 退出并停止 *Seafile Server*
- 打开文件浏览器，找到 Seafile 安装目录 `seafile-server-3.x.x`
- 右击 **gc.bat**, 并选择 以管理员身份运行

垃圾回收程序会运行并删除所有未用的数据块。

部署 Seafile 专业版

安装

- [推荐: 用脚本一键在 Ubuntu 16.04 或 CentOS 7 上安装专业版](#)
- [下载与安装 Seafile 专业版服务器](#)
- [从 Seafile 社区版服务器迁移到专业版服务器](#)
- [升级 Seafile 专业版服务器](#)
- [使用 Oracle 数据库部署](#)
- [Seafile 集群部署](#)

S3/Swift/Ceph

- [安装 Seafile 专业版服务器并使用亚马逊 S3](#)
- [安装 Seafile 专业版服务器并使用 Ceph](#)
- [安装 Seafile 专业版服务器并使用阿里云 OSS](#)

配置选项

- [Seafile 专业版服务器可配置的选项](#)

搜索功能

- [关于文件搜索的一些细节](#)

FAQ

- [Seafile 专业版服务器的 FAQ](#)

- [准备工作](#)
- [下载与安装](#)

准备工作

安装依赖库。

Ubuntu 16.04，可用以下命令安装全部依赖。

```
sudo apt-get install openjdk-8-jre poppler-utils libpython2.7 py
thon-pip \
mysql-server python-setuptools python-imaging python-mysqldb pyt
hon-memcache python-ldap \
python-urllib3

sudo pip install boto requests
sudo ln -sf /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java /usr/
bin/
```

CentOS 7 下:

```
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
sudo yum install java-1.7.0-openjdk poppler-utils python-setupto
ols \
python-imaging MySQL-python mariadb-server python-memcached pyth
on-ldap \
python-urllib3
sudo pip install boto
sudo /etc/init.d/mysqld start
```

补充说明：关于 Java

注意：Seafile 专业版需要 java 1.7 以上版本，请用 `java -version` 命令查看您系统中的默认 java 版本。如果不是 java 7，那么，请 [更新默认 java 版本](#)。

下载与安装 Seafile 专业版服务器

获得许可证书

将您得到的许可证书放在顶层目录下。比如，在这篇文档页面里，我们把 `/data/haiwen/` 作为顶层目录。

下载与解压 Seafile 专业版服务器

```
tar xf seafile-pro-server_2.1.5_x86-64.tar.gz
```

现在您的目录结构应该像如下这样：

```
haiwen
├── seafile-license.txt
└── seafile-pro-server-2.1.5/
```

安装

Seafile 专业版服务器的安装步骤与 Seafile 社区版服务器安装步骤相同。

1. 下载与安装 [Seafile 服务器](#) 并使用 [MySQL 数据库](#)
2. 使用 [Nginx](#) 为 Web 服务器
3. 配置和使用 [Memcached](#) (可选，建议用户数超过 50 人的时候配置)
4. 配置和使用 [HTTPS](#) (可选)
5. 配置开机启动脚本 (可选)

在您成功安装 Seafile 专业版服务器之后，您的目录结构应该像如下这样：

```
#tree haiwen -L 2
haiwen
├── seafile-license.txt # license file
├── conf                # configuration files
│   ├── ccnet.conf
│   ├── seafile.conf
│   ├── seahub_settings.py
│   └── seafdav.conf
├── ccnet
│   ├── mykey.peer
│   ├── PeerMgr
│   └── seafile.ini
├── pro-data            # data specific for professional version
│   └── seafevents.conf
├── seafile-data
├── seafile-pro-server-2.1.5
│   ├── reset-admin.sh
│   ├── runtime
│   ├── seafile
│   ├── seafile.sh
│   ├── seahub
│   ├── seahub-extra
│   ├── seahub.sh
│   ├── setup-seafile.sh
│   ├── setup-seafile-mysql.py
│   ├── setup-seafile-mysql.sh
│   └── upgrade
├── seahub-data
│   └── avatars         # for user avatars
```

- [限制条件](#)
- [准备工作](#)
- [迁移](#)
- [切换回社区版服务器](#)

限制条件

您可能已经部署过 **Seafile** 社区版服务器，并想要切换到[专业版](#)，或者反过来从专业版迁移到社区版。但是有一些限制条件需要您注意：

- 您只能在相同大版本的社区版服务器和专业版服务器之间进行切换。

这意味着，如果您正在使用 2.0 版本的社区版服务器，并且想要切换到 2.1 版本的专业版服务器，您必须先将您的社区版服务器升级到 2.1 版本，然后按照以下指南切换到 2.1 版本的专业版服务器。(版本号 2.1.x 中的最后一位没有关系)

准备工作

安装 **Java** 运行时环境 (JRE)

如果您的系统环境是 Ubuntu 或者 Debian，执行以下命令：

```
sudo apt-get install openjdk-7-jre
```

如果您的系统环境是 CentOS 或者 Red Hat，执行以下命令：

```
sudo yum install java-1.7.0-openjdk
```

注意：您也可以使用 Oracle JRE.

注意：**Seafile** 专业版需要 java 1.7 以上版本，请用 `java -version` 命令查看您系统中的默认 java 版本. 如果不是 java 7, 那么, 请 [更新默认 java 版本](#).

安装 **poppler-utils**

poppler-utils 提供对 pdf 文件的全文检索功能。

如果您的系统环境是 Ubuntu 或者 Debian，执行以下命令：

```
sudo apt-get install poppler-utils
```

如果您的系统环境是 CentOS 或者 Red Hat，执行以下命令：

```
sudo yum install poppler-utils
```

安装 Libreoffice 和 UNO 库

Libreoffice 和 Python-uno 库提供对办公文件的在线预览功能。如果它们没有安装，办公文件就不能在线预览。

如果您的系统环境是 Ubuntu 或者 Debian，执行以下命令：

```
sudo apt-get install libreoffice python-uno
```

如果您的系统环境是 CentOS 或者 RHEL，执行以下命令：

```
sudo yum install libreoffice libreoffice-headless libreoffice-py  
uno
```

对于其他的 Linux 发行版您可以参考：[Linux 下 LibreOffice 的安装](#)

一般地，您还需要为您的使用语言安装字体，特别是在亚洲地区，否则 office 文件和 pdf 文件不能正确地显示。

比如，中国的用户可能希望安装文泉驿系列的 TrueType 字体：

```
# 如果您的系统环境是 Ubuntu 或者 Debian，执行以下命令：  
sudo apt-get install ttf-wqy-microhei ttf-wqy-zenhei xfonts-wqy
```

迁移

我们假定您已经在 `/data/haiwen/seafile-server-2.1.0` 目录下部署了 Seafile 社区版服务器的 2.1.0 版本。

获得许可证书

从社区版迁移至专业版

将您获得的许可证书放在 **Seafile** 安装位置的顶层目录下。在我们的例子中，顶层目录是 `/data/haiwen/`。

下载与解压 **Seafile** 专业版服务器

- 32 位
- 64 位

您应该将压缩包解压到您的 **Seafile** 安装位置的顶层目录，在我们的例子中，顶层目录是 `/data/haiwen`。

```
tar xf seafile-pro-server_2.1.0_x86-64.tar.gz
```

现在您的目录结构像如下这样：

```
haiwen
├─ seafile-license.txt
├─ seafile-pro-server-2.1.0/
├─ seafile-server-2.1.0/
├─ ccnet/
├─ seafile-data/
├─ seahub-data/
├─ seahub.db
└─ seahub_settings.py
```

您应该已经注意到社区版服务器和专业版服务器名字的不同。以 64 位的 2.1.0 版本为例：

- **Seafile** 社区版服务器压缩包叫作 `seafile-server_2.1.0_x86-86.tar.gz`；解压后，文件夹名叫作 `seafile-server-2.1.0`
- **Seafile** 专业版服务器压缩包叫作 `seafile-pro-server_2.1.0_x86-86.tar.gz`；解压后，文件夹名叫作 `seafile-pro-server-2.1.0`

迁移

- 如果 **Seafile** 社区版服务器正在运行，请先停止它：

```
cd haiwen/seafile-server-2.1.0
./seafile.sh stop
./seahub.sh stop
```

- 运行迁移脚本

```
cd haiwen/seafile-pro-server-2.1.0/
./pro/pro.py setup --migrate
```

迁移脚本将会为您做以下的工作：

- 确保您满足所有的先决条件
- 创建必要的额外配置选项
- 更新 **avatar** 目录
- 创建额外的数据库表

现在您的目录结构像如下这样：

```
haiwen
├── seafile-license.txt
├── seafile-pro-server-2.1.0/
├── seafile-server-2.1.0/
├── ccnet/
├── seafile-data/
├── seahub-data/
├── seahub.db
├── seahub_settings.py
└── pro-data/
```

启用 **Seafile** 专业版服务器

```
cd haiwen/seafile-pro-server-2.1.0
./seafile.sh start
./seahub.sh start
```

切换回社区版服务器

如果 **Seafile** 专业版服务器正在运行，请先停止它：


```
cd haiwen/seafiler-pro-server-2.1.0/  
./seafiler.sh stop  
./seahub.sh stop
```

更新 **avatar** 目录的链接，参考[小版本升级](#)

```
cd haiwen/seafiler-server-2.1.0/  
./upgrade/minor-upgrade.sh
```

启用 **Seafiler** 社区版服务器

```
cd haiwen/seafiler-server-2.1.0/  
./seafiler.sh start  
./seahub.sh start
```

升级

升级

请参考 [升级](#)

使用 Oracle 数据库部署 Seafile 专业版服务器

- [安装依赖库](#)
- [数据库配置](#)
- [下载与安装](#)
- [启动 Seafile 服务器](#)

安装依赖库

Ubuntu 14.04，可用以下命令安装全部依赖。

```
sudo apt-get install openjdk-7-jre poppler-utils libpython2.7 py
thon-pip \
mysql-server python-setuptools python-imaging python-memcache py
thon-dev \
python-ldap python-urllib3

sudo pip install boto
```

CentOS 7 下:

```
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
sudo yum install java-1.7.0-openjdk poppler-utils python-setupto
ols \
python-imaging python-memcached python-ldap \
python-urllib3 python-devel gcc
sudo pip install boto
```

补充说明：关于 **Java**

注意：Seafile 专业版需要 java 1.7 以上版本，请用 `java -version` 命令查看您系统中的默认 java 版本。如果不是 java 7，那么，请 [更新默认 java 版本](#)。

安装 Oracle 客户端库

从 [Oracle 官网](#) 下载 `basic`，`sqlplus`，`devel` 三个 rpm 包。

在 CentOS/RedHat 下，

```
sudo rpm -ivh oracle-instantclient12.1-basic-12.1.0.2.0-1.x86_64
.rpm
sudo rpm -ivh oracle-instantclient12.1-sqlplus-12.1.0.2.0-1.x86_
64.rpm
sudo rpm -ivh oracle-instantclient12.1-devel-12.1.0.2.0-1.x86_64
.rpm
```

在 Ubuntu 下，使用 `alien` 程序来安装 rpm 包。

```
sudo apt-get install alien
sudo alien -i oracle-instantclient12.1-basic-12.1.0.2.0-1.x86_64
.rpm
sudo alien -i oracle-instantclient12.1-sqlplus-12.1.0.2.0-1.x86_
64.rpm
sudo alien -i oracle-instantclient12.1-devel-12.1.0.2.0-1.x86_64
.rpm
```

安装之后，相应的文件所在路径：

- 库：`/usr/lib/oracle/12.1/client64/lib`
- 客户端程序：`/usr/lib/oracle/12.1/client64/bin`

另外，在 `/usr/bin` 下面还有一个 `sqlplus64` 的符号链接指向 `/usr/lib/oracle/12.1/client64/bin/sqlplus`。

把 Oracle 库的路径加入 `LD_LIBRARY_PATH`：

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/oracle/12.1/cli
ent64/lib
```

设置 `ORACLE_HOME` 环境变量：

```
export ORACLE_HOME=/usr/lib/oracle/12.1/client64
```

把上面两个配置加入到 `~/.bashrc` 里面，以便重新登录之后仍然生效。

测试 `sqlplus` 客户端是否能工作：

```
sqlplus64 seafile/seafile@192.168.1.178/XE
```

上述命令通过 **seafile** 用户和密码访问 Oracle 数据库。命令的格式为 `sqlplus64 {user}/{password}@{server_address}/{service_name}`。 `service_name` 是 Oracle 数据库在安装的时候设置的，可以咨询 DBA。

最后安装 Oracle python 客户端库 `cx_Oracle`：

```
sudo pip install cx_Oracle
```

Oracle 数据库配置

推荐为 **Seafile** 服务器专门创建一个 Oracle 数据库用户和相应的 **Tablespace**，以便于管理。以下命令均在 **SQLPlus** 命令行里面执行。

创建给用户使用的 **tablespace**。该 **tablespace** 从 20M 开始，自动按需扩大。DBA 可以根据自己的需求和经验调整命令的参数。

```
create tablespace seafile_ts datafile 'seafile_ts.dat' size 20M
autoextend on;
```

创建用户（用户名 **seafile**，示例密码也是 **seafile**）并给予相应的权限，并关联 **tablespace**，限定最多使用 5GB 空间。**Seafile** 中有的表格会随着使用时间的增长而增长，比如用户的 **session** 表等，为了减少 **tablespace** 达到空间上限导致服务中断的可能性，我们建议给 **tablespace** 分配重组的空间。这些较大的表格可以定期清理，以减少空间使用。

```
create user seafile identified by seafile default tablespace sea
file_ts quota 5000M on seafile_ts;
```

赋予新用户权限。**Seafile** 服务器需要使用创建 **Sequence** 对象的权限。

```
grant connect, create table, create sequence, create trigger to
seafile;
```

下载与安装 **Seafile** 专业版服务器

获得许可证书

将您得到的许可证书放在顶层目录下。比如，在这篇文档页面里，我们把

```
/data/haiwen/
```

 作为顶层目录。

下载与解压 Seafile 专业版服务器

```
tar xf seafile-pro-server_2.1.5_x86-64.tar.gz
```

现在您的目录结构应该像如下这样：

```
haiwen
├── seafile-license.txt
└── seafile-pro-server-2.1.5/
```

安装

我们先按照使用 SQLite 数据库的方式来完成安装，然后再手工配置使用 Oracle 数据库。

```
cd seafile-server-*
./setup-seafile.sh #运行安装脚本并回答预设问题
```

如果你的系统中没有安装上面的某个软件，那么 Seafile 初始化脚本会提醒你安装相应的软件包。该脚本会依次询问你一些问题，从而一步步引导你配置 Seafile 的各项参数。

| 参数 | 作用 | 说明 |
|---------------------------------------|--|---|
| seafiler server name | seafiler 服务器的名字，目前该配置已经不再使用 | 3 ~ 15 个字符，可以用英文字母，数字，下划线 |
| seafiler server ip or domain | seafiler 服务器的 IP 地址或者域名 | 客户端将通过这个 IP 或者地址来访问你的 Seafiler 服务 |
| seafiler data dir | seafiler 数据存放的目录，用上面的例子，默认将是 /data/haiwen/seafiler-data | seafiler 数据将随着使用而逐渐增加，请把它放在一个有足够大空闲空间的分区上 |
| seafiler fileserver port | seafiler fileserver 使用的 TCP 端口 | 一般使用默认的 8082 端口，如果已经被占用，可以设置为其他的端口 |

如果安装正确完成，会打印成功消息

现在你的目录结构将会是如下:

```
#tree haiwen -L 2
haiwen
├── conf                                # configuration files
│   ├── ccnet.conf
│   ├── seafile.conf
│   ├── seahub_settings.py
│   └── seafdav.conf
├── ccnet
│   ├── mykey.peer
│   ├── PeerMgr
│   └── seafile.ini
├── installed
│   └── seafile-server_1.4.0_x86-64.tar.gz
├── seafile-data
├── seafile-server-1.4.0               # active version
│   ├── reset-admin.sh
│   ├── runtime
│   ├── seafile
│   ├── seafile.sh
│   ├── seahub
│   ├── seahub.sh
│   ├── setup-seafile.sh
│   └── upgrade
├── seafile-server-latest              # symbolic link to seafile-server-1.4
.0
├── seahub-data
│   └── avatars
└── seahub.db
```

`seafile-server-latest` 文件夹是当前 **Seafile** 服务器文件夹的符号链接.将来你升级到新版本后, 升级脚本会自动更新使其始终指向最新的 **Seafile** 服务器文件夹。

配置 **Seafile** 使用 **Oracle** 数据库

修改 `haiwen/conf/ccnet.conf` , 加入以下选项：


```
[Database]
ENGINE = oracle
HOST = 192.168.1.178
USER = seafile
PASSWD = seafile
SERVICE_NAME = XE
```

把 `HOST` , `USER` , `PASSWD` 替换成你的环境中具体的值, `SERVICE_NAME` 是 Oracle 数据库实例的 `service name`, 具体请咨询 DBA 并替换成你们的实际名称。

修改 `haiwen/conf/seafile.conf` , 加入以下选项:

```
[database]
type = oracle
host = 192.168.1.178
user = seafile
password = seafile
service_name = XE
```

修改 `haiwen/conf/seahub_settings.py` , 加入以下选项:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.oracle',
        'NAME': 'xe',
        'USER': 'seafile',
        'PASSWORD': 'seafile',
        'HOST': '192.168.1.178',
        'PORT': '1521',
    },
    'OPTIONS': {
        'threaded': True,
    },
}
```

修改 `haiwen/conf/seafevents.conf` , 加入以下选项:

```
[DATABASE]
type = oracle
host = 192.168.1.178
username = seafile
password = seafile
service_name = XE
```

在 **Oracle** 数据库中创建 **Seafile** 所需表格

在启动 seafile/ccnet 之前，需要先手工创建表格。创建表格的 SQL 语句在

seafile-server-latest/create-db/oracle 目录下面的 ccnet_db.sql ,
seafile_db.sql , seahub_db.sql 脚本里面。

你可以在 SQLPlus 命令行里面通过如下命令创建表格：

```
SQL> @seafile_db.sql
SQL> @ccnet_db.sql
SQL> @seahub_db.sql
```

启动 **Seafile** 服务器

启动 **Seafile** 前，需要先删除一些 **Seafile** 自带的文件

在 seafile-server-latest 目录下，运行如下命令

```
rm seafile/lib/libcIntsh*
```

启动 **Seafile** 服务器和 **Seahub** 网站

在 seafile-server-latest 目录下，运行如下命令

- 启动 Seafile:

```
./seafile.sh start # 启动 Seafile 服务
```

- 启动 Seahub

```
./seahub.sh start <port> # 启动 Seahub 网站（默认运行在8000端口上）
```

小贴士: 你第一次启动 seahub 时, seahub.sh 脚本会提示你创建一个 Seafile 管理员帐号。

服务启动后, 打开浏览器并输入以下地址

```
http://192.168.1.111:8000/
```

你会被重定向到登陆页面. 输入管理员用户名和密码即可。

恭喜! 现在你已经成功的安装了 Seafile 服务器。

在另一端口上运行 Seahub

如果你不想在默认的 8000 端口上运行 Seahub, 而是想自定义端口（比如8001）中运行, 请按以下步骤操作:

- 关闭 Seafile 服务器

```
./seahub.sh stop # 停止 Seafile 进程  
./seafile.sh stop # 停止 Seahub
```

- 更改 haiwen/conf/ccnet.conf 文件中 SERVICE_URL 的值(假设你的 ip 或者域名是 192.168.1.100), 如下 (从 5.0 版本开始, 可以直接在管理员界面中设置。注意, 如果同时在 Web 界面和配置文件中设置了这个值, 以 Web 界面的配置为准。):

```
SERVICE_URL = http://192.168.1.100:8001
```

- 重启 Seafile 服务器

```
./seafile.sh start # 启动 Seafile 服务  
./seahub.sh start 8001 # 启动 Seahub 网站（运行在8001端口上）
```

关闭/重启 Seafile 和 Seahub

关闭

```
./seahub.sh stop # 停止 Seahub  
./seafiler.sh stop # 停止 Seafiler 进程
```

重启

```
./seafiler.sh restart # 停止当前的 Seafiler 进程，然后重启 Seafiler  
./seahub.sh restart # 停止当前的 Seahub 进程，并在 8000 端口重新启动 Seahub
```

如果停止/重启的脚本运行失败

大多数情况下 `seafiler.sh seahub.sh` 脚本可以正常工作。如果遇到问题：

- 使用 **pgrep** 命令检查 `seafiler/seahub` 进程是否还在运行中

```
pgrep -f seafiler-controller # 查看 Seafiler 进程  
pgrep -f "seahub" # 查看 Seahub 进程
```

- 使用 **pkill** 命令杀掉相关进程

```
pkill -f seafiler-controller # 结束 Seafiler 进程  
pkill -f "seahub" # 结束 Seafiler 进程
```

一键安装脚本下额外配置

如果你使用 [一键安装脚本](#) 部署的 Seafiler，在完成上述配置之后，还需：

Ubuntu 16.04 root 用户下

在 `/etc/init.d/seafiler-server` 文件中，在 `fastcgi_port=8000` 与 `case "$1" in` 之间加入

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/oracle/12.1/client64/lib  
export ORACLE_HOME=/usr/lib/oracle/12.1/client64
```

Centos 7 root 用户下

1. 在 `/etc/systemd/system/seafile.service` 的 `[Service]` 配置中加入

```
Environment="LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/oracle/12.1/client64/lib"
Environment="ORACLE_HOME=/usr/lib/oracle/12.1/client64"
```

2. 在 `/etc/systemd/system/seahub.service` 的 `[Service]` 配置中加入

```
Environment="LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/oracle/12.1/client64/lib"
Environment="ORACLE_HOME=/usr/lib/oracle/12.1/client64"
```

3. 运行

```
systemctl enable seafile
```

OK!

查看seafile更多信息请访问:

- [Nginx 下配置 Seahub / Apache 下配置 Seahub](#)
- [Nginx 下启用 Hhttps / Apache 下启用 Hhttps](#)
- [Seafile LDAP配置](#)
- [管理员手册](#)

Seafile 企业版 LDAP 和 Active Directory 配置

LDAP (Light-weight Directory Access Protocol) 是企业广泛部署的用户信息管理服务，微软的活动目录服务器（Active Directory）完全兼容 LDAP。这个文档假定您已经了解了 LDAP 相关的知识和术语。

Seafile 是如何管理 LDAP 用户的

Seafile 中的用户分为两类：

- 保存在 Seafile 内部数据库中的用户。这些用户关联了一些属性，比如是否管理员，是否已激活等。这类用户又分为两个子类别：
 - 系统管理员直接创建的用户。这些用户保存在 ccnet 数据库里面的 EmailUser 表中。
 - 由 LDAP 导入的用户。当 LDAP 里的用户第一次登录 Seafile 时，Seafile 会把该用户的信息导入到内部数据库。
- 在 LDAP 中存在的用户。管理员可以通过配置文件指定 LDAP 中可以使用 Seafile 服务的用户范围。这些用户在第一次登录时被导入到 Seafile 数据库中。Seafile 只会直接操作存在数据库中的用户。

Seafile 会自动从内部数据库和 LDAP 中查找用户，只要用户存在于任何一个来源，他们都能登录。

基本的 LDAP/AD 集成配置

Seafile 要求 LDAP/AD 服务器中每个用户都有一个唯一的 ID。这个 ID 只能使用邮箱地址格式。一般来说，AD 有两个用户属性可以用作 Seafile 用户的 ID：

- email 地址：一般的机构都会给每个成员分配唯一的 email 地址，所以这是最常见配置。
- UserPrincipalName (UPN)：这个 AD 赋予给每个用户的一个唯一 ID。它的格式为 用户登录名@域名。尽管这个不是真实的 email 地址，但是它也能作为 Seafile 的用户 ID。如果机构的用户没有 email 地址，可以使用这个属性。

与 AD 集成

把下面的配置添加到 `ccnet.conf` 中。

如果你使用 email 地址作为用户唯一 ID：

```
[LDAP]
HOST = ldap://192.168.1.123/
BASE = cn=users,dc=example,dc=com
USER_DN = administrator@example.local
PASSWORD = secret
LOGIN_ATTR = mail
```

如果你使用 `UserPrincipalName` 作为用户 ID：

```
[LDAP]
HOST = ldap://192.168.1.123/
BASE = cn=users,dc=example,dc=com
USER_DN = administrator@example.local
PASSWORD = secret
LOGIN_ATTR = userPrincipalName
```

各个配置选项的含义如下：

- **HOST:** LDAP 服务器的地址 URL。如果您的 LDAP 服务器监听在非标准端口上，您也可以在 URL 里面包含端口号，如 `ldap://ldap.example.com:389`。
- **BASE:** 在 LDAP 服务器的组织架构中，用于查询用户的根节点的唯一名称（Distinguished Name，简称 DN）。这个节点下面的所有用户都可以访问 Seafile。注意这里必须使用 **OU** 或 **CN**，即 `BASE = cn=users,dc=example,dc=com` 或者 `BASE = ou=users,dc=example,dc=com` 可以工作。但是 `BASE = dc=example,dc=com` 不能工作。`BASE` 中可以填多个 OU，用 `;` 分隔。
- **USER_DN:** 用于查询 LDAP 服务器中信息的用户的 DN。这个用户应该有足够的权限访问 `BASE` 以下的信息。通常建议使用 LDAP/AD 的管理员用户。
- **PASSWORD:** `USER_DN` 对应的用户的密码。
- **LOGIN_ATTR:** 用作 Seafile 中用户登录 ID 的 LDAP 属性，可以使用 `mail` 或者 `userPrincipalName`。

注意：如果配置项包含中文，需要确保配置文件使用 **UTF8** 编码保存。

关于如何选定 `BASE` 和 `USER_DN` 的一些技巧：

- 要确定您的 `BASE` 属性，您首先需要打开域管理器的图形界面，并浏览您的组

织架构。

- 如果您想要让系统中所有用户都能够访问 Seafile，您可以使用 `'cn=users,dc=yourdomain,dc=com'` 作为 BASE 选项（需要替换成你们的域名）。
- 如果您只想要某个部门的人能访问，您可以把范围限定在某个 OU（Organization Unit）中。您可以使用 `dsquery` 命令行工具来查找相应 OU 的 DN。比如，如果 OU 的名字是 `'staffs'`，您可以运行 `dsquery ou -name staff`。更多的信息可以参考[这里](#)。
- AD 支持使用 `'user@domain.com'` 格式的用户名作为 `USER_DN`。比如您可以使用 `administrator@example.com` 作为 `USER_DN`。有些时候 AD 不能正确识别这种格式。此时您可以使用 `dsquery` 来查找用户的 DN。比如，假设用户名是 `'seafileuser'`，运行 `dsquery user -name seafileuser` 来找到该用户的 DN。更多的信息可以参考[这里](#)。

与其他 LDAP 服务器集成

把以下配置添加到 `ccnet.conf` 中：

```
[LDAP]
HOST = ldap://192.168.1.123/
BASE = ou=users,dc=example,dc=com
USER_DN = cn=admin,dc=example,dc=com
PASSWORD = secret
LOGIN_ATTR = mail
```

配置选项的含义与 AD 配置相同。不过，你只能使用 `mail` 作为用户的 ID，因为其他 LDAP 服务器不支持 `UserPrincipalName`。

测试你的 LDAP 配置

从专业版 5.0.0 开始，我们提供了一个测试你的 LDAP 配置合法性的命令行工具。

使用这个工具之前，请先确定你使用的是专业版而且 Linux 系统中安装了

`python-ldap` 这个包。

```
sudo apt-get install python-ldap
```

然后你可以执行测试：


```
cd seafile-server-latest
./pro/pro.py ldapsync --test
```

测试脚本会检查 `ccnet.conf` 里面 `[LDAP]` 下面的配置。如果一切正常工作，它会打印出搜索的前十个用户。如果出错，它会打印出可能出错的配置信息。

注意当前这个脚本并不支持测试 `[LDAP_SYNC]` 下面的 LDAP 同步配置。

重启 Seafile 服务

在更新了 `ccnet.conf` 之后，你必须重启 **Seafile** 服务以使得配置生效。

LDAP 高级配置选项

使用多个 BASE DN

当您想把公司中多个 OU 加入 Seafile 中时，您可以使用在配置中指定多个 BASE DN。您可以在"BASE"配置中指定一个 DN 的列表，标识名由";"分开，比如：

```
cn=developers,dc=example,dc=com;cn=marketing,dc=example,dc=com
```

用户过滤选项

当你的公司组织庞大，但是只有一小部分人使用 Seafile 的时候，搜索过滤器（Search filter）会很有用处。过滤器可以通过修改"FILTER"配置来实现，例如，在 LDAP 配置中增加以下语句：

```
FILTER = memberOf=CN=group,CN=developers,DC=example,DC=com
```

请注意上面的示例只是象征性的简介。 `memberOf` 只有在活动目录(Active Directory)中才适用。

把 Seafile 用户限定在 AD 的一个组中

您可以利用用户过滤器选项来只允许 AD 某个组中的用户使用 Seafile。

1. 首先，您需要找到这个组的 DN。我们再次使用 `dsquery` 命令。比如，如果组的名称是 'seafilegroup'，那么您可以运行 `dsquery group -name`

seafilegroup 。

2. 然后您可以把一下配置加入 ccnet.conf 的 LDAP 配置中：

```
FILTER = memberOf={dsquery 命令的输出}
```

使用结果分页扩展（**paged results extension**）

LDAP 协议 v3 支持一个称为 "paged results" 的扩展功能。当您在 LDAP 中有大量用户的时候，这个选项能够大大提高列出用户的速度。而且，AD 限制了单次请求中返回的用户条目数量，您需要启用这个选项才能避免查询错误。

在 Seafile 企业版中，在 LDAP 配置中加入以下设置：

```
USE_PAGED_RESULT = true
```

【可选】配置 **AD** 用户同步

在企业版中，你还可以把 AD 中用户的其他信息导入到 Seafile 的内部数据库。这些信息包括：

- 用户的全名，部门等。这些信息可以用户方便地根据人名+部门来查找用户，在共享文件的时候比较有用。
- 用户的 Windows 登录名。导入到数据库之后，用户可以直接使用 Windows 用户名来登录 Seafile。
- 当用户在 AD 中被删除之后（比如离职），Seafile 会自动禁用他的账户。

AD 用户同步配置

把以下配置添加到 ccnet.conf 里：

```
[LDAP]
.....

[LDAP_SYNC]
ENABLE_USER_SYNC = true
DEACTIVE_USER_IF_NOTFOUND = true
SYNC_INTERVAL = 60
USER_OBJECT_CLASS = person
ENABLE_EXTRA_USER_INFO_SYNC = true
FIRST_NAME_ATTR = givenName
LAST_NAME_ATTR = sn
USER_NAME_REVERSE = true
DEPT_ATTR = department
UID_ATTR = sAMAccountName
ACTIVATE_USER_WHEN_IMPORT = true
```

各个选项的含义：

- **ENABLE_USER_SYNC**: 设置为 true 以启用用户同步功能
- **DEACTIVE_USER_IF_NOTFOUND**: 设置为 true 以禁用已经在 AD 中删除的用户
- **SYNC_INTERVAL**: 以分钟为单位的同步间隔。默认为 60 分钟同步一次。
- **USER_OBJECT_CLASS**: 用户对象的 class 名字。在 AD 中一般是 "person"。默认值也是 "person"。
- **ENABLE_EXTRA_USER_INFO_SYNC**: 同步用户的额外信息，包括用户的全名，部门，Windows 登录名。
- **FIRST_NAME_ATTR**: 用户名字对应的属性，默认使用 "givenName" 属性。
- **LAST_NAME_ATTR**: 用户的姓氏属性。默认使用 "sn" 属性。
- **USER_NAME_REVERSE**: 中文的人名里面姓氏和名字与西方的习惯相反，所以对中文名字，需要把这个选项设置为 true。
- **DEPT_ATTR**: 用户的部门属性。默认使用 "department" 属性。
- **UID_ATTR**: 用户的 Windows 登录名属性。一般使用 "sAMAccountName" 属性。
- **ACTIVATE_USER_WHEN_IMPORT**: 导入用户之后是否立即激活，默认是 true，即立即激活该用户。

如果你选择了 "userPrincipalName" 作为用户的唯一 ID，Seafile 不能使用这个 ID 作为 email 地址来发送通知邮件给用户。如果你的 AD 中也有用户的 email 地址属性，你可以把这个属性同步到 Seafile 的内部数据库中。配置的选项是：

- **CONTACT_EMAIL_ATTR**: 一般来说你可以把它设置为 "mail" 属性。

手工执行 **AD** 同步

在配置完成后，你可以手工执行同步来测试配置是否有效。

```
cd seafile-server-latest
./pro/pro.py ldapsync
```

让 **AD** 同步不要自动导入新用户到数据库中

在默认情况下，AD 同步会把 AD 中检测到的新用户自动同步到 Seafile 的数据库中。这些新创建的用户会被自动设置为“已激活”。这些用户会被算入 license 用户数量中。

我们考虑以下场景：你在 AD 中有很多用户。但是你不想要一次购买足够多的 license 把所有用户一次全部加入 Seafile。此时自动导入新用户的功能就会很容易把你的 license 消耗完毕，导致系统不可用。解决方案是：新用户只有在第一次登录的时候创建，而 AD 同步不会自动创建新用户。我们提供了一个选项来满足这种需求：

```
[LDAP_SYNC]
IMPORT_NEW_USER = false
```

【可选】导入 **AD** 中的群组到 **Seafile**

请参考[英文的文档](#)

同步 AD 群组

在 4.1.0 版本之后，专业版开始支持从 LDAP 或者 AD 导入(同步)群组到 seafile。

工作原理

导入或同步的过程是从 LDAP 服务器上的组映射到 seafile 的内部数据库中的组。这个过程是单向的。

- 数据库中对组的任何改变都不会回传到 LDAP；
- 除了“设置成员为组管理员”之外，数据库中对组的任何更改将被下一个LDAP同步操作覆盖。如果要添加或删除成员则只能在LDAP服务器上执行此操作。
- 导入组的创建者将会被设置为系统管理员。

一些LDAP服务器(如ad)允许将组设置为另一个组的成员。这被称为“嵌套”。我们的程序支持同步嵌套组。假设B组是A组成员，结果将是：B组的每个成员均为A组和B组的成员。

有两种运作模式：

- 周期：同步过程将在固定的时间间隔内执行；
- 手动：可以通过执行一个脚本来触发同步过程；

前提条件

您已经在系统中安装了 python-ldap 库。

在 Debian 或 Ubuntu 下：

```
sudo apt-get install python-ldap
```

在 CentOS 或 RedHat 下

```
sudo yum install python-ldap
```

配置

在启用 LDAP 组同步之前，您应该已经配置好 LDAP 身份验证。有关详细信息请参考[Seafile 企业版 LDAP 和 Active Directory 配置](#)。

以下是 LDAP 组同步想相关选项。它们定义在 `ccnet.conf` 的“[LDAP_SYNC]”配置段中。

- **ENABLE_GROUP_SYNC**：如果要启用 LDAP 组同步请设置为“true”。
- **SYNC_INTERVAL**：同步周期，单位是分钟，默认设置为60分钟。
- **GROUP_OBJECT_CLASS**：这是用于搜索组对象的类的名称。在 Active directory 中，它通常是"group";在OpenLDAP或其他中，可以使用"groupOfNames","groupOfUniqueNames" 或者 "posixGroup",这取决于你使用的LDAP服务器。默认设置为"group"。
- **GROUP_FILTER**：在搜索组对象时使用的附加筛选器。如果设置了，最终用于搜索的筛选器是"(&(objectClass=GROUP_OBJECT_CLASS)(GROUP_FILTER))"，否则使用的筛选器将是"(objectClass=GROUP_OBJECT_CLASS)"。
- **GROUP_MEMBER_ATTR**：在加载组的成员时使用的属性字段。对于大多数 directory 服务器，属性是“member”，这也是默认值。对于"posixGroup"，它应该被设置为"memberUid"。
- **USER_ATTR_IN_MEMBERUID**：“memberuid”选项中的用户属性集,用于“posixgroup”。默认值为“uid”。

组的搜索基础是 `ccnet.conf` 中设置在"[LDAP]"配置段的"BASE_DN"。

这有一个关于 Active Directory 的配置示例：

```
[LDAP]
HOST = ldap://192.168.1.123/
BASE = cn=users,dc=example,dc=com
USER_DN = administrator@example.local
PASSWORD = secret
LOGIN_ATTR = mail

[LDAP_SYNC]
ENABLE_GROUP_SYNC = true
SYNC_INTERVAL = 60
```

对于AD，除了"ENABLE_GROUP_SYNC"之外，通常不需要配置其他选项。因为其他选项的默认值是AD的常用值。如果LDAP服务器中有特殊设置，则只设置相应的选项。

这有一个关于 OpenLDAP 的配置示例：

```
[LDAP]
HOST = ldap://192.168.1.123/
BASE = ou=users,dc=example,dc=com
USER_DN = cn=admin,dc=example,dc=com
PASSWORD = secret
LOGIN_ATTR = mail

[LDAP_SYNC]
ENABLE_GROUP_SYNC = true
SYNC_INTERVAL = 60
GROUP_OBJECT_CLASS = groupOfNames
```

注意 在您重启 **seafile** 服务器后，不会立即同步，它在第一次同步周期后进行同步。例如，如果将同步周期设置为30分钟，则第一次自动同步将在您重启后的30分钟后发生。要立即同步，您需要手动触发。下一节将介绍这一情况。

运行同步后，您应该在日志 `logs/seafevents` 中看到如下所示的日志信息。并且在系统管理页面中应该能看到那些组。

```
[2015-03-30 18:15:05,109] [DEBUG] create group 1, and add dn pair
CN=DnsUpdateProxy,CN=Users,DC=Seafile,DC=local<->1 success.
[2015-03-30 18:15:05,145] [DEBUG] create group 2, and add dn pair
CN=Domain Computers,CN=Users,DC=Seafile,DC=local<->2 success.
[2015-03-30 18:15:05,154] [DEBUG] create group 3, and add dn pair
CN=Domain Users,CN=Users,DC=Seafile,DC=local<->3 success.
[2015-03-30 18:15:05,164] [DEBUG] create group 4, and add dn pair
CN=Domain Admins,CN=Users,DC=Seafile,DC=local<->4 success.
[2015-03-30 18:15:05,176] [DEBUG] create group 5, and add dn pair
CN=RAS and IAS Servers,CN=Users,DC=Seafile,DC=local<->5 success.
[2015-03-30 18:15:05,186] [DEBUG] create group 6, and add dn pair
CN=Enterprise Admins,CN=Users,DC=Seafile,DC=local<->6 success.
[2015-03-30 18:15:05,197] [DEBUG] create group 7, and add dn pair
CN=dev,CN=Users,DC=Seafile,DC=local<->7 success.
```

手动触发同步

手动触发 LDAP 组同步。

```
cd seafile-server-latest  
./pro/pro.py ldapsync
```


要求

要想使用 ADFS 登陆到 Seafile，需要以下组件：

- 1、安装了 [ADFS](#) 的 windows 服务器。安装 ADFS 和相关配置详情请参考 [本文](#)。
- 2、对于 ADFS 服务器的 SSL 有效证书，在这里我们使用 **adfs-server.adfs.com** 作为域名示例。
- 3、对于 seafile 服务器的 SSL 有效证书，在这里我们使用 **demo.seafile.com** 作为域名示例。

准备证书文件

1、SP(Service Provider) 的 x.509 证书

可以通过以下方式获取：

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout sp.key -out sp.crt
```

x.509 证书用来签署和加密诸如 SAML 的 NameID 和 Metadata 等元素。

然后将这两个文件复制到 **/seahub-data/certs**。如果证书文件夹不存在，请创建它。

2、IdP(Identity Provider) 的 x.509 证书

1. 登陆到 ADFS 服务器并且打开 ADFS 管理。
2. 双击 **Service** 并选择 **Certificates**。
3. 导出 **Token-Signing** 证书：
 - i. 右击证书并选择 **View Certificate**。
 - ii. 选择 **Details** 选项卡。
 - iii. 单击 **Copy to File** (选择 **DER encoded binary X.509**)。
4. 将此证书转换为 PEM 格式，重命名为 **idp.crt**
5. 复制它到 **/seahub-data/certs**。

准备 IdP 元数据文件

1. 打开 <https://adfs-server.adfs.com/federationmetadata/2007-06/federationmetadata.xml>
2. 保存这个 xml 文件，重命名为 **idp_federation_metadata.xml**
3. 复制它到 **/seahub-data/certs**。

在 **seafile** 服务器上安装

- 对于 Ubuntu 16.04

```
sudo apt install libxmlsec1
sudo pip install cryptography djangosaml2
```

配置 **seafile**

添加以下配置到 **seahub_settings.py**

```
from os import path
import saml2
import saml2.saml

CERTS_DIR = '<seafile-install-path>/seahub-data/certs'
SP_SERVICE_URL = 'https://demo.seafile.com'
XMLSEC_BINARY = '/usr/local/bin/xmlsec1'
ATTRIBUTE_MAP_DIR = '<seafile-install-path>/seafile-server-latest/seahub-extra/seahub_extra/adfs_auth/attribute-maps'
SAML_ATTRIBUTE_MAPPING = {
    'DisplayName': ('display_name', ),
    'ContactEmail': ('contact_email', ),
    'Deparment': ('department', ),
    'Telephone': ('telephone', ),
}

ENABLE_ADFS_LOGIN = True
EXTRA_AUTHENTICATION_BACKENDS = (
    'seahub_extra.adfs_auth.backends.Saml2Backend',
)
SAML_USE_NAME_ID_AS_USERNAME = True
LOGIN_REDIRECT_URL = '/saml2/complete/'
SAML_CONFIG = {
```

```

# full path to the xmlsec1 binary programm
'xmlsec_binary': XMLSEC_BINARY,

'allow_unknown_attributes': True,

# your entity id, usually your subdomain plus the url to the
metadata view
'entityid': SP_SERVICE_URL + '/saml2/metadata/',

# directory with attribute mapping
'attribute_map_dir': ATTRIBUTE_MAP_DIR,

# this block states what services we provide
'service': {
    # we are just a lonely SP
    'sp' : {
        "allow_unsolicited": True,
        'name': 'Federated Seafile Service',
        'name_id_format': saml2.saml.NAMEID_FORMAT_EMAILADDR
ESS,
        'endpoints': {
            # url and binding to the assestion consumer servi
ce view
            # do not change the binding or service name
            'assertion_consumer_service': [
                (SP_SERVICE_URL + '/saml2/acs/',
                 saml2.BINDING_HTTP_POST),
            ],
            # url and binding to the single logout service v
iew
            # do not change the binding or service name
            'single_logout_service': [
                (SP_SERVICE_URL + '/saml2/ls/',
                 saml2.BINDING_HTTP_REDIRECT),
                (SP_SERVICE_URL + '/saml2/ls/post',
                 saml2.BINDING_HTTP_POST),
            ],
        },

        # attributes that this project need to identify a us
er
        'required_attributes': ["uid"],

```

```

        # attributes that may be useful to have but not required
        'optional_attributes': ['eduPersonAffiliation', ],

        # in this section the list of IdPs we talk to are defined
        'idp': {
            # we do not need a WAYF service since there is
            # only an IdP defined here. This IdP should be
            # present in our metadata

            # the keys of this dictionary are entity ids
            'https://ads-server.adfs.com/federationmetadata/2007-06/federationmetadata.xml': {
                'single_sign_on_service': {
                    saml2.BINDING_HTTP_REDIRECT: 'https://ads-server.adfs.com/adfs/ls/idpinitiatedsignon.aspx',
                },
                'single_logout_service': {
                    saml2.BINDING_HTTP_REDIRECT: 'https://ads-server.adfs.com/adfs/ls/?wa=wsignout1.0',
                },
            },
        },
    },

    # where the remote metadata is stored
    'metadata': {
        'local': [path.join(CERTS_DIR, 'idp_federation_metadata.xml')],
    },

    # set to 1 to output debugging information
    'debug': 1,

    # Signing
    'key_file': '',
    'cert_file': path.join(CERTS_DIR, 'certs/idp.crt'), # from IdP

```

```
# Encryption
'encryption_keypairs': [{
    'key_file': path.join(CERTS_DIR, 'certs/sp.key'), # private part
    'cert_file': path.join(CERTS_DIR, 'certs/sp.crt'), # public part
}],

'valid_for': 24, # how long is our metadata valid
}
```

配置 ADFS 服务

1. 添加 Relying Party Trust

Relying Party Trust 是 Seafire 和 ADFS 之间的连接。

- i. 登陆到 ADFS 服务器并打开 ADFS 管理界面。
- ii. 双击 **Trust Relationships**，然后右键 **Relying Party Trusts**，选择 **Add Relying Party Trust...**。
- iii. 选择 **Import data about the relying party published online or one a local network**，在 **Federation metadata address** 中输入
`https://demo.seafire.com/saml2/metadata/`
- iv. 然后 **Next** 直到 **Finish**。

2. 添加 Relying Party Claim Rules

Relying Party Claim Rules 是用于 windows 域中 seafire 和用户的通信。

Important：在 windows 域中的用户必须要设置了 **E-mail** 值。

- i. 右键点击 relying party trust 并且选择 **Edit Claim Rules...**。
- ii. 在 Issuance Transform Rules **Add Rules...**
- iii. 选择 **Send LDAP Attribute as Claims** 作为申请规则模版来用。
- iv. 给 claim 一个名称，例如：LDAP Attributes。
- v. 将 Attribute Store 设置为 **Active Directory**，LDAP Attribute 设置为 **E-Mail-Addresses**，Outgoing Claim Type 设置为 **E-mail Address**。

- vi. 选择 **Finish** 。
- vii. 再次单击 **Add Rule...** 。
- viii. 选择 **Transform an Incoming Claim** 。
- ix. 给它一个名字例如：**Email to Name ID** 。
- x. 输入的 claim 类型应该是 **E-mail Address** (它必须跟 rule #1 中的 Outgoing Claim Type 相匹配) 。
- xi. Outgoing claim 的类型是 **Name ID** (这是seafile配置策略中的要求
`'name_id_format': saml2.saml.NAMEID_FORMAT_EMAILADDRESS`) 。
- xii. Outgoing name ID 格式为 **Email** 。
- xiii. 通过所有的 **claim** 的值 并且单击 **Finish** 。

测试

重启服务后，你可以打开一个web浏览器并且输入 `https://demo.seafile.com` , 在登陆对话框中应该有一个 `adfs` 按钮。单击该按钮将重定向到 ADFS 服务器 (`adfs-server.adfs.com`),如果用户名密码正确，你将被重定向到seafile主页。

对于desktop客户端，只需要在"Add a new account"窗口点击"Shibboleth Login"，输入 `https://demo.seafile.com` ,单击 OK 按钮将会打开一个新的窗口显示 ADFS服务的登录页面，如果用户名和密码正确，窗口将关闭并显示seafile资料库面板。

-
- <https://support.zendesk.com/hc/en-us/articles/203663886-Setting-up-single-sign-on-using-Active-Directory-with-ADFS-and-SAML-Plus-and-Enterprise>
 - http://wiki.servicenow.com/?title=Configuring_ADFS_2.0_to_Communicate_with_SAML_2.0#gsc.tab=0
 - <https://github.com/rohe/pysaml2/blob/master/src/saml2/saml.py>

开启 Office/PDF 文件在线预览

Seafile 专业版服务器支持在线预览 office 文件，配置方法如下。

安装 Libreoffice/UNO

Office 预览依赖于 Libreoffice 4.1+ 和 Python-uno 库。

Ubuntu/Debian:

```
sudo apt-get install libreoffice libreoffice-script-provider-python
```

For older version of Ubuntu: `sudo apt-get install libreoffice python-uno`

Centos/RHEL:

```
sudo yum install libreoffice libreoffice-headless libreoffice-python-uno
```

其他 Linux 发行版可以参考: [Installation of LibreOffice on Linux](#)

你还需要安装字体文件:

```
# For ubuntu/debian
sudo apt-get install ttf-wqy-microhei ttf-wqy-zenhei xfonts-wqy
```

开启配置项

1. 打开 `conf/seafevents.conf` , 添加:

```
[OFFICE CONVERTER]
enabled = true
```

2. 保存后 `seafevents.conf` 重启 Seafile 服务 `./seafile.sh restart`

其他配置选项

[OFFICE CONVERTER]

```
## How many libreoffice worker processes to run concurrently
workers = 1

## where to store the converted office/pdf files. Default is /tmp/.
outputdir = /tmp/

## how many pages are allowed to be previewed online. Default is
50 pages
max-pages = 50

## the max size of documents to allow to be previewed online, in
MB. Default is 2 MB
## Preview a large file (for example >30M) online will freeze the
browser.
max-size = 2
```

FAQ

Office 预览不能工作，日志文件在哪？

你可以查看 `logs/seafevents.log`

Office 预览不能工作的一个可能原因是你的机器上的 `libreoffice` 版本太低，可以用以下方法修复

删除安装的 `libreoffice`:

```
sudo apt-get remove libreoffice* python-uno python3-uno
```

从官网下载最新版 [libreoffice official site](#)，并安装

```
tar xf LibreOffice_4.1.6_Linux_x86-64_deb.tar.gz
cd LibreOffice_4.1.6.2_Linux_x86-64_deb
cd DEBS
sudo dpkg -i *.deb
```


开启 Office/PDF 文件在线预览

重启 Seafile 服务

```
./seafile.sh restart
```

Office Web App (OWA) 集成

在Seafile专业版 4.4.0(或更高版本)中，可以使用 Microsoft Office Web App 在线预览文档。Office Web App 为所有的 Office 文档提供最佳预览。对于具有 Microsoft Office 批量许可证的组织，可以免费使用Office Web APP。有关 Office Web App 的更多信息以及如何部署Office Web App，请参阅

<https://technet.microsoft.com/en-us/library/jj219456.aspx>。

Seafile自身的Office文件预览仍然是默认的。使用 Office Web App 进行预览，请添加以下配置项到 `seahub_settings.py` 中。

```
# Enable Office Web App
ENABLE_OFFICE_WEB_APP = True

# Url of Office Web App's discovery page
# The discovery page tells Seafile how to interact with Office W
# eb App when view file online
# You should change `http://example.office-web-app.com` to your
# actual Office Web App server address
OFFICE_WEB_APP_BASE_URL = 'http://example.office-web-app.com/hos
ting/discovery'

# Expiration of WOPI access token
# WOPI access token is a string used by Seafile to determine the
# file's
# identity and permissions when use Office Web App view it onlin
# e
# And for security reason, this token should expire after a set
# time period
WOPI_ACCESS_TOKEN_EXPIRATION = 30 * 60 # seconds

# List of file formats that you want to view through Office Web
# App
# You can change this value according to your preferences
# And of course you should make sure your Office Web App support
# s to preview
# the files with the specified extensions
OFFICE_WEB_APP_FILE_EXTENSION = ('ods', 'xls', 'xlsb', 'xlsm', '
xlsx', 'ppsx', 'ppt',
    'pptm', 'pptx', 'doc', 'docm', 'docx')
```

```
# Enable edit files through Office Web App
ENABLE_OFFICE_WEB_APP_EDIT = True

# types of files should be editable through Office Web App
# Note, Office Web App 2016 is needed for editing docx
OFFICE_WEB_APP_EDIT_FILE_EXTENSION = ('xlsx', 'pptx', 'docx')

# HTTPS authentication related (optional)

# Server certificates
# Path to a CA_BUNDLE file or directory with certificates of trusted CAs
# NOTE: If set this setting to a directory, the directory must have been processed using the c_rehash utility supplied with OpenSSL.
OFFICE_WEB_APP_SERVER_CA = '/path/to/certfile'

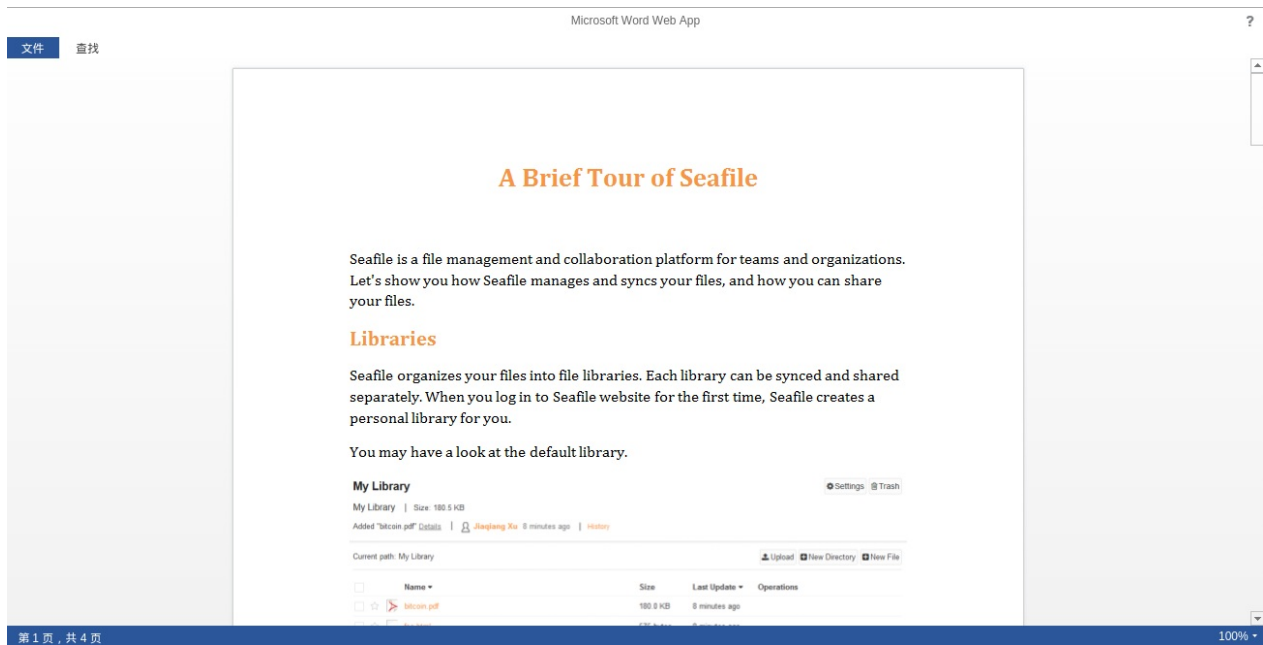
# Client certificates
# You can specify a single file (containing the private key and the certificate) to use as client side certificate
OFFICE_WEB_APP_CLIENT_PEM = 'path/to/client.pem'

# or you can specify these two file path to use as client side certificate
OFFICE_WEB_APP_CLIENT_CERT = 'path/to/client.cert'
OFFICE_WEB_APP_CLIENT_KEY = 'path/to/client.key'
```

然后重启服务

```
./seafile.sh restart
./seahub.sh restart
```

单击您在seahub_sttings.py中指定的文档后，您将看到新的预览页面。



故障排查

了解Web应用集成原理将帮助你排查问题。当用户访问页面时：

1. (seahub->浏览器) Seahub 将生成一个包含 iframe 的页面并将其发送到浏览器。
2. (浏览器->office web app) 使用 iframe，浏览器将尝试从 office web app 加载预览页面。
3. (office web app->seahub) office web app 接收请求并向 Seahub 发送请求以获取文件内容。
4. (office web app->浏览器) office web app 发送文件预览页面给浏览器。

请检查Seahub的Nginx日志（步骤3）和Office Web App，以查看哪个步骤出错。

- [FAQ](#)
 - 我不能搜索 **office/PDF** 文档
 - 当我搜索一个关键词时，没有返回结果
 - 不能搜索加密的文件

常见问题

我不能搜索 **office/PDF** 文档

首先，确认在 **seafevents.conf** 文件中，已经设置 `index_office_pdf = true`：

```
[INDEX FILES]
...
index_office_pdf = true
```

然后，检查是否安装 **pdftotext**

如果 **pdftotext** 没有安装，就不能从 PDF 文件中提取文本。

```
which pdftotext
```

执行上面的命令，如果没有输出，那么您需要安装它：

```
sudo apt-get install poppler-utils
```

pdftotext 安装完成后，您需要重新构建您的搜索索引。

```
./pro/pro.py search --clear
./pro/pro.py search --update
```

当我搜索一个关键词时，没有返回结果

默认情况下，搜索索引每 10 分钟更新一次。所以，在第一次索引更新前，无论您搜索什么都不会返回结果。

为了使搜索能够立即生效，您可以手动更新搜索索引：

文件搜索说明

- 确保您已经启动了 **Seafile** 服务器
- 手动更新搜索索引：

```
cd haiwen/seafile-pro-server-2.0.4  
./pro/pro.py search --update
```

不能搜索加密的文件

这是因为服务器不能索引加密的文件，因为它们被加密了。

文件病毒扫描

注意：自从 Seafile 5.0.0 版本以后，所有的配置文件转移到了统一的配置文件目录 **conf**。详情

在 Seafile 专业版 4.4.0(及以上)版本中，Seafile 可以在后台扫描上传文件中的恶意内容。配置为定期运行后，扫描程序将扫描所有现有库。此后的每次扫描中，该程序仅扫描上次扫描后新上传的或更新过的文件。对于每一个文件，该进程都执行一条用户指定的病毒扫描指令来检查文件是否是病毒。大多数反病毒程序都为 Linux 提供了命令行工具。

要启用该功能，添加以下配置项到 `seafile.conf`：

```
[virus_scan]
scan_command = (command for checking virus)
virus_code = (command exit codes when file is virus)
nonvirus_code = (command exit codes when file is not virus)
scan_interval = (scanning interval, in unit of minutes, default
to 60 minutes)
```

关于选项的更多细节：

- 在 Linux/Unix，大多数病毒扫描命令针对病毒或非病毒会返回特定的退出码。更多信息请参阅反病毒程序手册。

以下提供了 ClamAV (<http://www.clamav.net/>) 的示例：

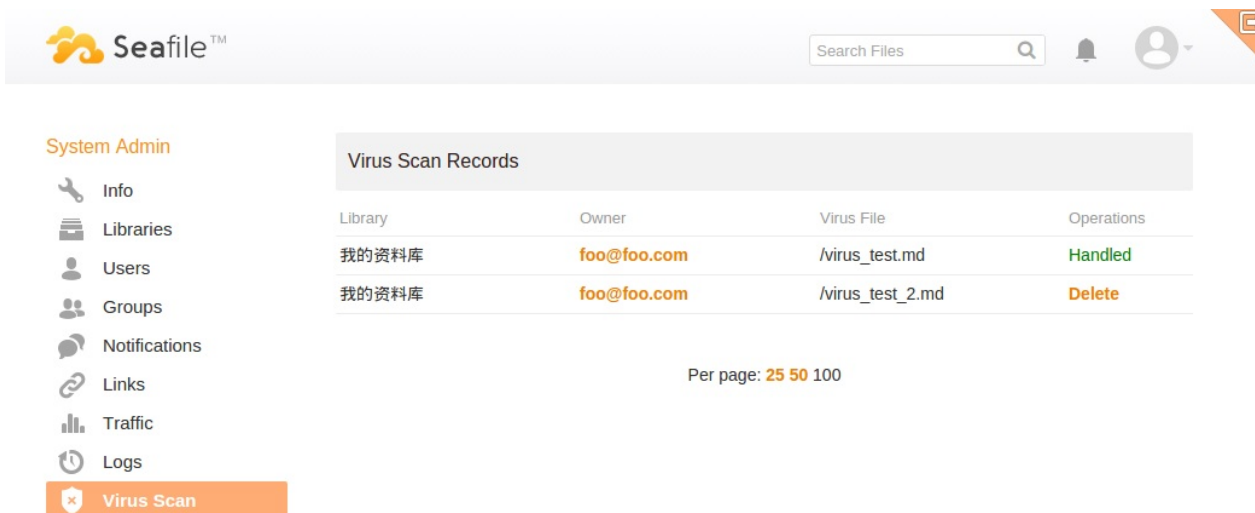
```
[virus_scan]
scan_command = clamscan
virus_code = 1
nonvirus_code = 0
```

要测试您的配置是否正常工作，您可以手动触发扫描：

```
cd seafile-server-latest
./pro/pro.py virus_scan
```

病毒扫描

如果检测到病毒，则可以在管理区域的“病毒扫描”页面上查看扫描记录并删除受感染文件。



在专业版 6.0.0 之后，添加更多的选项来提供更细粒度的病毒扫描控制。

```
[virus_scan]
.....
scan_size_limit = (size limit for files to be scanned)
scan_skip_ext = (a comma (',') separated list of file extensions
to be ignored)
threads = (number of concurrent threads for scan, one thread for
one file, default to 4)
```

文件扩展名应该以 ‘.’ 开头，扩展名不区分大小写。默认情况下，具有以下扩展名的文件将被忽略：

```
.bmp, .gif, .ico, .png, .jpg, .mp3, .mp4, .wav, .avi, .rmvb, .mkv
```

您提供的列表将覆盖默认列表。

卡巴斯基病毒扫描配置

请参考[英文文档](#)

Web 文件断点续传

在Web界面上传大文件时，如果网络不可靠，则可能会中断上传。如果上传可以从上次停止的地方恢复，将会很方便。在 Seafile 专业版 4.4.0及更新版本中，支持断点续传功能。

断点续传的工作原理如下：

1. 用户在Web界面上传一个大文件，并且在上传过程中连接中断。
2. 服务器将记住上传停止的位置。
3. 当同一个文件被上传到相同资料库中的同一文件夹时，服务器会告诉浏览器从哪里开始上传。

限制：

1. 只支持重新上传；文件更新和文件夹上传无法断点续传。
2. 只支持 Chrome, Firefox, IE 10+ 。

要启用断点续传功能，请添加如下配置到 `seahub_settings.py` 中：

```
ENABLE_RESUMABLE_FILEUPLOAD = True
```

在 Seafile 集群中，为了使此功能如期工作，必须执行以下两个特殊配置之一：

1. seafile-server-latest/seafile-data/httptemp 目录应该通过NFS共享给所有前端 Seafile 服务器。
2. 或者，将负载均衡器配置为始终将来自同一IP地址的请求发送到固定的后端服务器。

Amazon S3 下安装

Note: Seafile 服务器 5.0.0 之后，所有配置文件都移动到了统一的 **conf** 目录下。
[了解详情](#)。

准备工作

为了安装 Seafile 专业版服务器并使用亚马逊 S3，您需要：

- 按照 [下载安装 Seafile 专业版服务器](#) 指南安装基本的 Seafile 专业版服务器。
- 安装 python 的 `boto` 库。它可以用来访问 S3 服务。

```
sudo easy_install boto
```

- 安装和使用 Memcached. 为了提高性能，Seafile 会将部分小对象缓存在 memcached 里面。我们推荐给 memcached 分配128MB内存。修改 `/etc/memcached.conf`

```
# Start with a cap of 64 megs of memory. It's reasonable, and the daemon default
# Note that the daemon will grow to this size, but does not start out holding this much
# memory
# -m 64
-m 128
```

修改 seafile.conf

编辑 `/data/haiwen/conf/seafile.conf` 文件

```
[commit_object_backend]
name = s3
# bucket 的名字只能使用小写字母，数字，短划线
bucket = my-commit-objects
key_id = your-key-id
key = your-secret-key
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100

[fs_object_backend]
name = s3
# bucket 的名字只能使用小写字母，数字，短划线
bucket = my-fs-objects
key_id = your-key-id
key = your-secret-key
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100

[block_backend]
name = s3
# bucket 的名字只能使用小写字母，数字，短划线
bucket = my-block-objects
key_id = your-key-id
key = your-secret-key
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100
```

建议您为 **commit**，**fs** 和 **block objects** 分别创建 **buckets**。**key_id** 和 **key** 用来提供 S3 的身份认证。您可以在您的 AWS 账户页面的“安全证书”段找到 **key_id** 和 **key**。

当您在 S3 上创建 **buckets** 时，请先阅读 [S3 bucket 命名规则](#)。注意，尤其不要在 **bucket** 的名字中使用大写字母（不要使用骆驼式命名法，比如 **MyCommitObjects**）。

为了获得最佳性能，强烈建议您安装 **memcached** 并且为 **objects** 启用 **memcache**。

使用新的 **S3** 服务区

自2014年一月起，新的 AWS 服务区只对 S3 提供版本 4 的认证签名协议支持。这包括中国区。

要新的服务区使用 S3，在 `commit_object_backend`, `fs_object_backend`, `block_backend` 相关选项中加入一下额外的选项：

```
use_v4_signature = true
# eu-central-1 for Frankfurt region
aws_region = eu-central-1
```

为了让搜索等服务也能在新的 AWS 服务区工作，你还需要在 `~/.boto` 文件中加入一下几行：

```
[s3]
use-sigv4 = True
```

使用 memcached 集群

在集群环境中，你可能会使用多个 memcached 服务器组成一个集群。你需要在 `seafile.conf` 中列出所有服务器的地址。加入一下选项：

```
memcached_options = --SERVER=192.168.1.134 --SERVER=192.168.1.135 --SERVER=192.168.1.136 --POOL-MIN=10 --POOL-MAX=100 --RETRY-TIMEOUT=3600
```

注意最后有一个 `--RETRY-TIMEOUT=3600` 选项。这个选项对于处理 memcached 服务器宕机的情况非常重要。在一个 memcached 服务器宕机之后，Seafile 服务器会在 `RETRY-TIMEOUT` 秒之内不再尝试使用这个服务器。你需要把这个超时设置到一个相对较大的值，以防止由于频繁重试一个不可用服务器而导致经常给客户端返回错误。

使用与 S3 兼容的对象存储产品

目前已经有很多对象存储产品兼容 S3 的协议，比如 OpenStack Swift 和 Ceph 的 RGW。你可以通过以下配置使用 S3 兼容的对象存储：

```
[commit_object_backend]
name = s3
bucket = my-commit-objects
key_id = your-key-id
key = your-secret-key
host = 192.168.1.123:8080
path_style_request = true
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100

[fs_object_backend]
name = s3
bucket = my-fs-objects
key_id = your-key-id
key = your-secret-key
host = 192.168.1.123:8080
path_style_request = true
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100

[block_backend]
name = s3
bucket = my-block-objects
key_id = your-key-id
key = your-secret-key
host = 192.168.1.123:8080
path_style_request = true
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100
```

其中，`host` 是存储服务的地址加端口。你不能在前面添加 `http` 或者 `https` 选项。`path_style_request` 选项让 **Seafile** 使用形如

`https://192.168.1.123:8080/bucketname/object` 去访问对象。在亚马逊的 S3 服务中，默认的 URL 格式是虚拟主机格式，比如

`https://bucketname.s3.amazonaws.com/object`。但是一般的对象存储并不支持这种格式。

Ceph 下安装

Note: Seafiler 服务器 5.0.0 之后，所有配置文件都移动到了统一的 **conf** 目录下。
[了解详情](#).

Ceph 是一种可扩展的分布式存储系统。Seafiler 可以使用 Ceph 的 RADOS 对象存储层作为存储后端。

拷贝 **ceph** 的配置文件和客户端的密钥环

Seafiler 可以看作 Ceph/RADOS 的客户端，所以它需要访问 **ceph** 集群的配置文件和密钥环。您必须将 **ceph** 管理员节点 `/etc/ceph` 目录下的文件拷贝到 **seafiler** 的机器上。

```
seafiler-machine# sudo scp user@ceph-admin-node:/etc/ceph/ /etc
```

安装 **Python Ceph** 客户端库

WebDAV 访问和文件检索服务依赖于 `python ceph` 客户端库，因此需要在系统中安装。

在 Debian/Ubuntu 上

```
sudo apt-get install python-ceph
```

在 CentOS/RHEL 上

```
sudo yum install python-rados
```

编辑 **Seafiler** 配置文件

编辑 `seafiler.conf` 文件，添加以下几行：

```
[block_backend]
name = ceph
ceph_config = /etc/ceph/ceph.conf
pool = seafiler-blocks
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100

[commit_object_backend]
name = ceph
ceph_config = /etc/ceph/ceph.conf
pool = seafiler-commits
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100

[fs_object_backend]
name = ceph
ceph_config = /etc/ceph/ceph.conf
pool = seafiler-fs
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100
```

建议您为 commit，fs 和 block objects 分别创建连接池：

```
ceph-admin-node# rados mkpool seafiler-blocks
ceph-admin-node# rados mkpool seafiler-commits
ceph-admin-node# rados mkpool seafiler-fs
```

删除安装包中自带的 C++ 库

由于 Seafiler 的发布包是在比较老版本的 CentOS 上面编译的，所以自带的 C++ 库以及 ceph 客户端库和新的操作系统不兼容。因此，在比较新的系统上（比如 CentOS 7, Ubuntu 16.04）需要把发布包里面几个自带的库删除掉，以保证 seafiler 会使用安装在系统中的库。

```
cd seafiler-server-latest/seafiler/lib
rm librados.so.2 libstdc++.so.6 libnspr4.so
```

安装并启用 memcached

为了最佳性能，强烈建议您安装 memcached 并为 objects 启用 memcache。

我们建议您为 memcached 分配 128MB 的内存空间。编辑 /etc/memcached.conf 文件如下：

```
# Start with a cap of 64 megs of memory. It's reasonable, and the daemon default
# Note that the daemon will grow to this size, but does not start out holding this much
# memory
# -m 64
-m 128
```

使用 memcached 集群

在集群环境中，你可能会使用多个 memcached 服务器组成一个集群。你需要在 seafile.conf 中列出所有服务器的地址。加入一下选项：

```
memcached_options = --SERVER=192.168.1.134 --SERVER=192.168.1.135 --SERVER=192.168.1.136 --POOL-MIN=10 --POOL-MAX=100 --RETRY-TIMEOUT=3600
```

注意最后有一个 `--RETRY-TIMEOUT=3600` 选项。这个选项对于处理 memcached 服务器宕机的情况非常重要。在一个 memcached 服务器宕机之后，Seafile 服务器会在 `RETRY-TIMEOUT` 秒之内不再尝试使用这个服务器。你需要把这个超时设置到一个相对较大的值，以防止由于频繁重试一个不可用服务器而导致经常给客户端返回错误。

使用其他 Ceph 用户

上述的配置会使用默认的 Ceph 用户（client.admin）来访问 Ceph。从安全性的角度考虑，你可能想使用其他用户来为 Seafile 提供 Ceph 访问。假设你创建的 Ceph 用户 id 为 seafile，加入以下配置：

```
[block_backend]
name = ceph
ceph_config = /etc/ceph/ceph.conf
# Sepcify Ceph user for Seafile here
ceph_client_id = seafile
pool = seafile-blocks
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100

[commit_object_backend]
name = ceph
ceph_config = /etc/ceph/ceph.conf
# Sepcify Ceph user for Seafile here
ceph_client_id = seafile
pool = seafile-commits
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100

[fs_object_backend]
name = ceph
ceph_config = /etc/ceph/ceph.conf
# Sepcify Ceph user for Seafile here
ceph_client_id = seafile
pool = seafile-fs
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100
```

你可以通过以下命令创建 ceph 用户：

```
ceph auth add client.seafile \
    mds 'allow' \
    mon 'allow r' \
    osd 'allow rwx pool=seafile-blocks, allow rwx pool=seafile-com
mits, allow rwx pool=seafile-fs'
```

你还需要把这个用户的 keyring 路径写入 /etc/ceph/ceph.conf 中：

```
[client.seafile]
keyring = <path to user's keyring file>
```


使用阿里云开放存储（**OSS**）作为后端存储

准备工作

为了安装 Seafile 专业版服务器并使用阿里云OSS，您需要：

- 按照 [下载安装 Seafile 专业版服务器](#) 指南安装基本的 Seafile 专业版服务器。
- 安装 oss2 软件包：`sudo pip install oss2`，更多安装帮助可以参考[这个文档](#)。
- 安装和使用 Memcached。Seafile 会将部分对象缓存在 memcached 中，以提高性能。建议至少给 memcached 分配 128MB 内存。请修改 memcached 的配置文件（Ubuntu 上在 /etc/memcached.conf）：

```
# Start with a cap of 64 megs of memory. It's reasonable, and the daemon default
# Note that the daemon will grow to this size, but does not start out holding this much
# memory
# -m 64
-m 128
```

修改 seafile.conf

编辑 `/data/haiwen/conf/seafile.conf` 文件，添加下面几行：

```
[commit_object_backend]
name = oss
bucket = <your-seafile-commits-bucket>
key_id = <your-key-id>
key = <your-key>
region = beijing
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100

[fs_object_backend]
name = oss
bucket = <your-seafile-fs-bucket>
key_id = <your-key-id>
key = <your-key>
region = beijing
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100

[block_backend]
name = oss
bucket = <your-seafile-blocks-bucket>
key_id = <your-key-id>
key = <your-key>
region = beijing
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=
100
```

关于上面配置的几点说明：

- 建议您为 **commit**，**fs** 和 **block objects** 分别创建 **buckets**。为了性能和节省网络流量成本，您应该在 **seafile** 服务器运行的区域内创建 **bucket**。
- **key_id** 和 **key** 用来提供 **OSS** 的身份认证。您可以在 **OSS** 管理界面上找到它们。
- **region** 是您创建的 **bucket** 所在的区域，比如 **beijing**, **hangzhou**, **shenzhen** 等。

在 VPC 中使用 OSS

在 6.0.9 版本之前，Seafile 仅支持使用经典网络环境下的 OSS 服务。VPC（虚拟私有网络）环境下的 OSS 服务地址不同于经典网络，因此需要在配置环境中指定 OSS 访问地址。6.0.9 版本后开始支持配置 OSS 访问地址，从而实现了对 VPC OSS 服务的支持。

使用如下的配置：

```
[commit_object_backend]
name = oss
bucket = <your-seafile-commits-bucket>
key_id = <your-key-id>
key = <your-key>
endpoint = vpc100-oss-cn-beijing.aliyuncs.com
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=100

[fs_object_backend]
name = oss
bucket = <your-seafile-fs-bucket>
key_id = <your-key-id>
key = <your-key>
endpoint = vpc100-oss-cn-beijing.aliyuncs.com
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=100

[block_backend]
name = oss
bucket = <your-seafile-blocks-bucket>
key_id = <your-key-id>
key = <your-key>
endpoint = vpc100-oss-cn-beijing.aliyuncs.com
memcached_options = --SERVER=localhost --POOL-MIN=10 --POOL-MAX=100
```

与经典网络下的配置相比，上述配置使用 `endpoint` 选项替换了 `region` 选项。相应的 `endpoint` 地址可以在 https://help.aliyun.com/document_detail/31837.html 上面找到。

`endpoint` 是一个通用选项，你也可以把它设置为经典网络下的 OSS 访问地址，一样可以工作。

使用 memcached 集群

在集群环境中，你可能会使用多个 memcached 服务器组成一个集群。你需要在 seafile.conf 中列出所有服务器的地址。加入一下选项：

```
memcached_options = --SERVER=192.168.1.134 --SERVER=192.168.1.135 --SERVER=192.168.1.136 --POOL-MIN=10 --POOL-MAX=100 --RETRY-TIMEOUT=3600
```

注意最后有一个 `--RETRY-TIMEOUT=3600` 选项。这个选项对于处理 memcached 服务器宕机的情况非常重要。在一个 memcached 服务器宕机之后，Seafile 服务器会在 `RETRY-TIMEOUT` 秒之内不再尝试使用这个服务器。你需要把这个超时设置到一个相对较大的值，以防止由于频繁重试一个不可用服务器而导致经常给客户端返回错误。

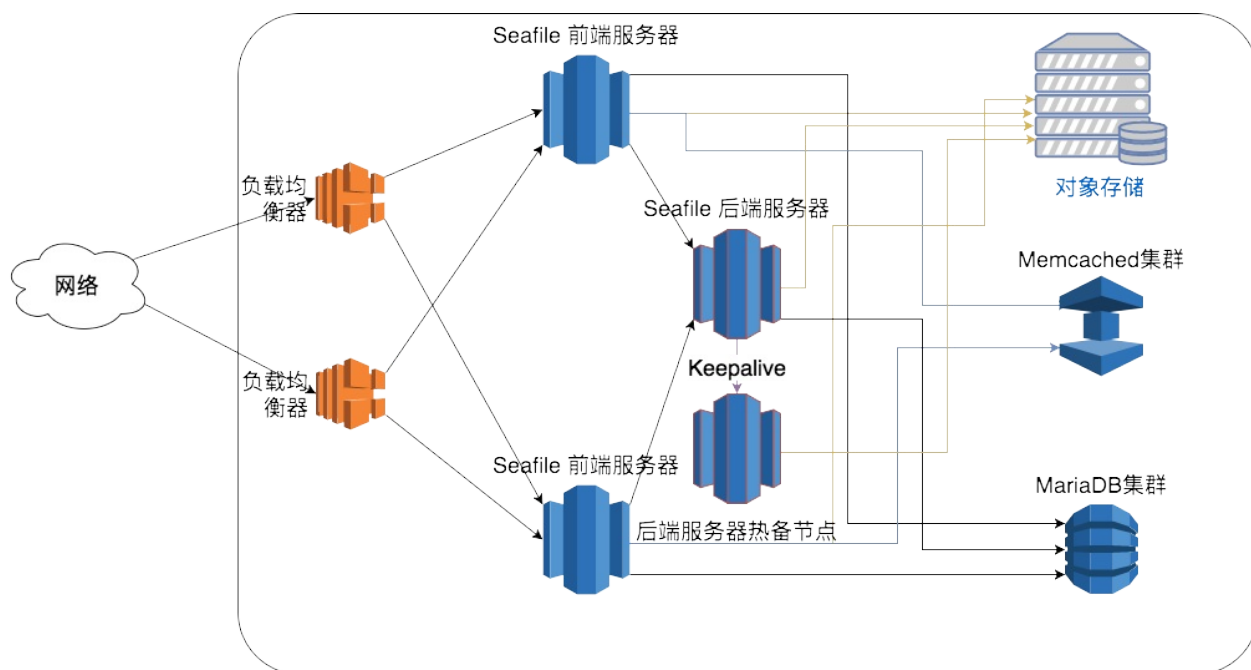
集群部署

架构

Seafile 集群方案采用了3层架构：

- 负载均衡层：将接入的流量分配到 **seafile** 服务器上。并且可以通过部署多个负载均衡器来实现高可用。
- **Seafile** 服务器集群：一组 **seafile** 服务器实例。如果集群中的某个实例不可用，负载均衡调度器将停止向其传输流量，以实现高可用。
- 后端存储、数据库、缓存

该架构支持横向扩展。这意味着，您可以通过添加更多的 **Seafile** 服务器来分担处理流量，架构图如下：



负载均衡器可以使用硬件负载均衡器或者 HAProxy 这种软件负载均衡器。当使用 HAProxy 作为负载均衡器时，可以采用主从热备的架构。客户端通过一个虚拟 IP 地址（VIP）来访问 Seafile 服务。该 VIP 平时绑定在 HAProxy 主节点上。主从节点之间通过 **Keepalived** 来进行可用性监控，当从节点检测到主节点不可用时，主动将 VIP 接管过来。对客户端来说，这个切换是不可感知的。

Seafile 服务器集群中的服务器分为两类角色：前端服务器节点和后端服务器节点。前端服务器直接为客户端提供文件访问的服务，包括网页、移动端和文件同步访问。后端服务器节点负责运行一些后台任务，包括文件全文检索、Office 文件预览生成器、AD 用户信息同步等。一个集群中可以有任意多个的前端服务器节点，可

根据性能需求进行扩展。而后端服务器节点只有一个，不过由于其运行的时候后台任务，不可用并不会影响主要的文件访问功能，所以可以不配置高可用。如果需要为后端节点实现高可用，可以通过主从热备的方式，用 **Keepalived** 来实现主后端节点不可用时，自动切换到从节点。

Seafile 前端服务器节点上有两个主要组件：**web 服务**(Nginx/Apache)和 **Seafile 应用程序服务**。**web 服务**将请求从客户端传递到 **seafile 应用程序服务**。**Seafile 服务器**独立工作，它们不知道对方的状态。这意味着集群中某个服务器发生故障不会影响到其他服务器实例。负载均衡服务器负责检测故障和重新分发请求。

即便 **Seafile 服务器**是独立工作的，它们也必须共享一些会话信息。所有共享的会话信息被保存在 **memcached 服务**上。因此，所有的 **Seafile 服务器**必须连接到相同的 **memcached 服务器**（集群）。

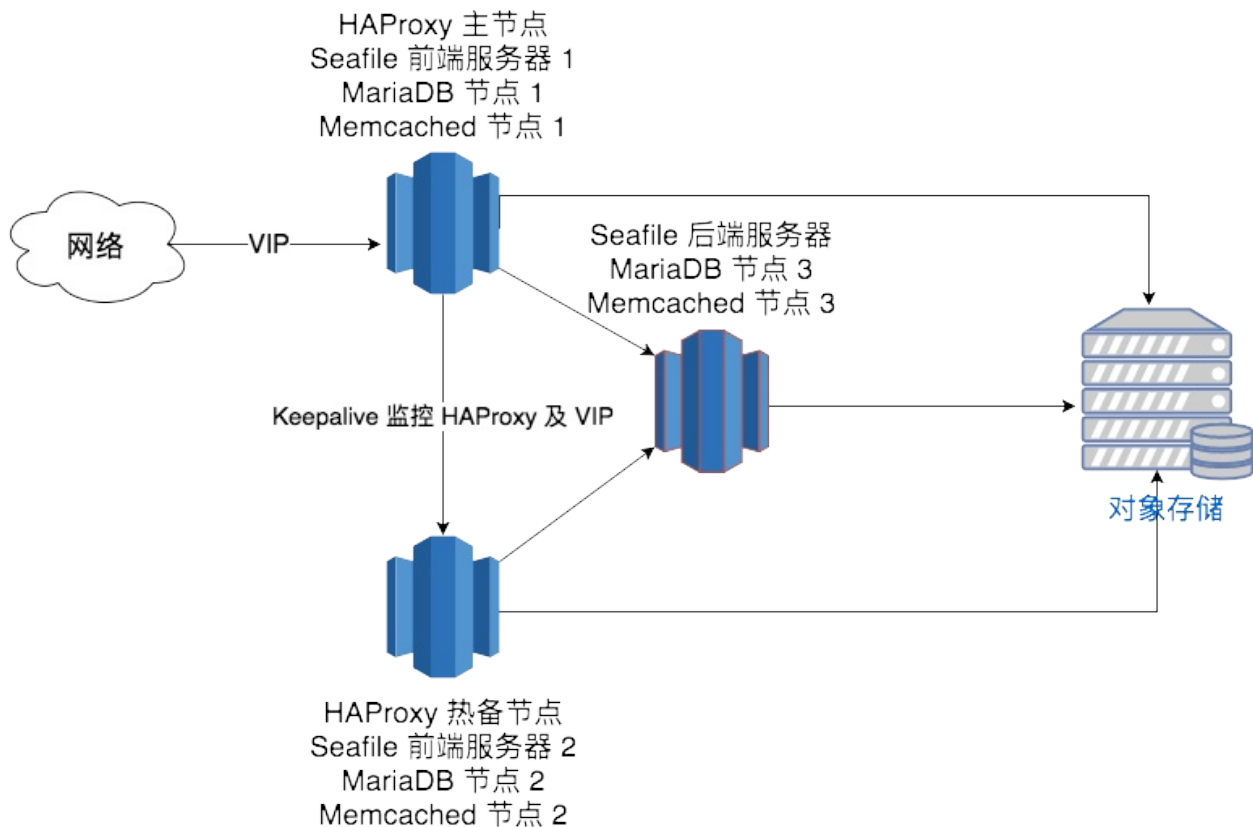
所有的 **Seafile 服务器**都访问相同的用户数据。这些用户数据包括两部分：一部分存储在 **MariaDB 数据库**中，另一部分存储在后端分布式存储集群中 (**S3, Ceph etc.**)。所有的应用程序服务器都向客户提供同样的数据。所有的应用服务器都必须能够连接到相同的数据库或相同的数据库集群。

从整个服务的高可用性上考虑，我们建议把 **MariaDB** 和 **Memcached** 都配置成集群模式，并且 **MariaDB** 和 **Memcached** 可以部署在相同的服务器上，以节省硬件资源。具体的配置方式可参考[这个文档](#)。

上述的集群架构是针对相对大型的集群而设计的，集群中各个部件一般都运行在独立的服务器上。按照上述的架构，我们需要的服务器资源为：

- 负载均衡：2台服务器
- **Seafile 前端服务器**集群：至少2台服务器
- **Seafile 后端服务器**：1台服务器，如果配置高可用需要2台服务器
- **Memcached + MariaDB 集群**：3台服务器
- 文件存储：取决于使用的存储后端类型，如果使用分布式存储，也需要多台服务器构成集群

有时为了节约硬件资源，我们可以把多个部件部署在一个服务器上。我们可以利用最少3台服务器来实现 **Seafile 集群**。架构图如下：



上述3节点的架构中，我们把负载均衡和 Seafiler 前端服务器共享到两台服务器上，第三台服务器作为后端服务器节点。而 MariaDB 和 Memcached 的集群则部署到这3台服务器上。文件存储后端并没有算入这个架构的服务器数量中。

部署 Seafiler 集群有以下几个步骤：

1. 准备硬件、操作系统、memcached和数据库
2. 部署单个 Seafiler 服务器节点（使用seafiler安装脚本）
3. 将单节点配置复制到其他 Seafiler 服务器节点
4. 配置后端服务
5. 部署负载均衡服务器

注意，这个安装部署步骤假设 memcached, MariaDB 以及分布式存储都是独立于 Seafiler 集群之外的。如果您需要使用 3 节点的最小部署架构，可以根据这个步骤进行简化。

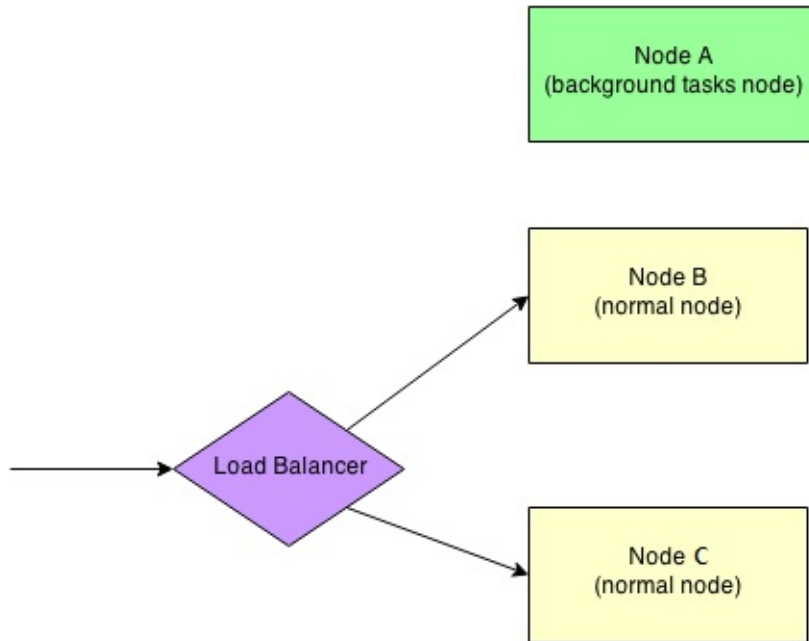
准备工作

硬件、存储、memcached、数据库

至少3台Linux服务器，至少4核，8GB内存。

假如一个seafiler集群中有三个节点：A、B 和 C

- 节点 A 作为后端节点，用来执行后端任务
- 节点 B 和 C 作为前端节点，用来接收来自客户端的请求



memcached、MariaDB 集群的配置可参考[这个文档](#)。

安装 Python 依赖库

在每个节点上,您需要安装一些python库。

首先确定您已经安装了 **Python 2.7**，然后执行：

```
sudo easy_install pip
sudo pip install boto
```

如果您收到了一条错误信息 "Wheel installs require setuptools >= ...",在上边的pip和boto之间运行它：

```
sudo pip install setuptools --no-use-wheel --upgrade
```

部署一个单节点 Seafile 服务器

要确保每一个 Seafile 服务器上的配置文件是一致的。不要在每台机器上分别配置 **seafile** 服务器,这非常重要。您应该在一台计算机上配置好**seafile**服务器,然后将配置目录复制到其他计算机。

安装 Seafile

Seafile 专业版的安装过程和社区版相同，建议您使用 Seafile 安装脚本。

Seafile 安装脚本

安装脚本可以帮助您快速的安装好 Seafile 服务器，并配置好 MariaDB, Memcached, WebDAV, Ngnix 和开机自动启动脚本。

使用步骤

安装干净的 Ubuntu 14.04, 16.04 或 CentOS 7 系统，并做好镜像 (如果安装失败需要还原到镜像)。

切换到 root 账号 (sudo -i)

获取安装脚本

Ubuntu 14.04:

```
wget https://raw.githubusercontent.com/haiwen/seafile-server-installer-cn/master/seafile-server-ubuntu-14-04-amd64-http
```

Ubuntu 16.04 (适用于 6.0.0 及以上版本) :

```
wget https://raw.githubusercontent.com/haiwen/seafile-server-installer-cn/master/seafile-server-ubuntu-16-04-amd64-http
```

CentOS 7:

```
wget https://raw.githubusercontent.com/haiwen/seafile-server-installer-cn/master/seafile-server-centos-7-amd64-http
```

运行安装脚本并指定要安装的版本 (比如 6.0.10)

Ubuntu :

```
bash seafile-server-ubuntu-14-04-amd64-http 6.0.10
```

CentOS 7 :

```
bash seafile-server-centos-7-amd64-http 6.0.10
```

脚本会让你选择要安装的版本, 按照提示进行选择即可:(专业版安装请选择'2')

- 要安装专业版, 需要先将下载好的专业版的包 `seafile-pro-server_VERSION_x86-64.tar.gz` 放到 `/opt/` 目录下

该脚本运行完后会在命令行中打印配置信息和管理员账号密码, 请仔细阅读。(你也可以查看安装日志 `/opt/seafile/aio_seafile-server.log`), MySQL 密码在 `/root/.my.cnf` 中。

访问测试

访问 `http:// <IP of node>`, 输入安装完成时命令行中返回的管理员账号密码, 即可访问。

登录管理后台—设置页面, 修改 `SERVICE_URL` 和 `FILE_SERVER_ROOT` 为当前地址, 否则无法上传文件。

至此单机 Seafile 服务已经安装完成, 数据库为本地mysql, 文件数据存放在本地磁盘目录: `/opt/seafile/seafile-data`

配置为集群服务

要作为集群部署, 还需要对配置文件做一些额外配置。

配置 `seafile.conf`

您需要添加以下配置信息到 `seafile.conf`

```
[cluster]
enabled = true
memcached_options = --SERVER=<IP of memcached node> --POOL-MIN=1
0 --POOL-MAX=100
```

如果您部署了一个memcached集群, 您需要添加所有的memcached服务器地址到 `seafile.conf`, 格式如下:

```
[cluster]
enabled = true
memcached_options = --SERVER=<IP of memcached node1> --SERVER=<I
P of memcached node2> --SERVER=<IP of memcached node3> --POOL-MI
N=10 --POOL-MAX=100 --RETRY-TIMEOUT=3600
```

注意，上面的配置中有一个 `--RETRY-TIMEOUT=3600` 选项，此选项对于处理 memcached 服务器故障非常重要。在集群中的 memcached 服务器发生故障后，seafile 服务器将在 `--RETRY-TIMEOUT` 定义的时间内（以秒为单位）停止尝试使用它。此超时设置应该较长时间，以防止 seafile 频繁重试失败的服务器，这可能导致客户端经常请求错误。

（可选）Seafile 服务器也可打开一个指定端口作为负载均衡器做健康状况检测时使用。默认情况下，seafile 使用端口 11001。可以通过向 `seafile.conf` 添加以下配置来更改此配置项：

```
[cluster]
health_check_port = 12345
```

配置 `seahub_settings.py`

若此前已经配置好使用 memcached 服务，还需要添加以下配置项到

`seahub_setting.py`。该配置指明 Seahub 将用户头像保存在数据库中并缓存到 memcached，还要将 css 缓存到本地内存中。如何配置 memcached 请参考 ["使用 memcached"](#)：

```
AVATAR_FILE_STORAGE = 'seahub.base.database_storage.DatabaseStor
age'

COMPRESS_CACHE_BACKEND = 'django.core.cache.backends.locmem.LocM
emCache'
```

配置 `seafevents.conf`

在 `seafevents.conf` 中添加以下内容以禁用本地服务器上的文件索引服务，因为文件索引服务应该在专用后台服务器上启动。

```
[INDEX FILES]
external_es_server = true
```

以下是 `[INDEX FILES]` 配置段的部分示例：

```
[INDEX FILES]
enabled = true
interval = 10m
index_office_pdf = true
external_es_server = true
es_host = background.seafile.com
es_port = 9200
```

注意：`enabled = true` 应该保持不变。`es_host = <IP of background node>` 指定后端服务器地址。

切换为远程数据库

首先停止seafile服务

需要在远程数据库里创建3个数据库

```
create database `ccnet_db` character set = 'utf8';
create database `seafile_db` character set = 'utf8';
create database `seahub_db` character set = 'utf8';
```

创建seafile用户，并授权访问以上数据库

```
create user 'seafile'@' ' identified by 'seafile';
GRANT ALL PRIVILEGES ON `ccnet_db`.* to 'seafile'@' ';
GRANT ALL PRIVILEGES ON `seafile_db`.* to 'seafile'@' ';
GRANT ALL PRIVILEGES ON `seahub_db`.* to 'seafile'@' ';
```

导入 `seahub` 的数据表，数据表在单机安装目录下 `seafile-server-laster/seahub/sql/mysql.sql`

另外还需要在`seahub_db`里新增一张数据表：

```
CREATE TABLE `avatar_uploaded` (`filename` TEXT NOT NULL, `filename_md5` CHAR(32) NOT NULL PRIMARY KEY, `data` MEDIUMTEXT NOT NULL, `size` INTEGER NOT NULL, `mtime` datetime NOT NULL);
```

修改**conf**目录下的配置

停止**seaf**，进入**conf**目录（默认在顶层安装目录下）

`vim ccnet.conf`，修改**[Database]**配置段如下：

```
[Database]
ENGINE = mysql
HOST = <IP of mysql node>
PORT = 3306
USER = seaf
PASSWD = seaf
DB = ccnet_db
CONNECTIONT_CHARSET = utf8
```

`vim seaf.conf`，修改**[database]**配置段如下：

```
[database]
type = mysql
host = <IP of mysql node>
port = 3306
user = seaf
password = seaf
db_name = seaf_db
connection_charset = utf8
```

`vim seahub_settings.py`，修改相关数据库配置如下：


```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'seahub_db',
        'USER': 'seafile',
        'PASSWORD': 'seafile',
        'HOST': '<IP of mysql node>',
        'PORT': '3306'
    }
}
```

`vim seafevents.conf` ,修改[DATABASE]配置段如下：

```
[DATABASE]
type = mysql
host = <IP of mysql node>
port = 3306
username = seafile
password = seafile
name = seahub_db
```

再启动seafile服务，调用命令 `seafile-server-latest/reset-admin.sh` 可重置新的管理员账号。

配置后端存储

还需要将后端云存储系统的设置添加到配置文件中，这里提供了4种后端存储的配置方案：

- 使用 NFS：[NFS 下集群安装](#)
- 使用 S3：[Amazon S3 下安装](#)
- 使用 OSS：[使用阿里云OSS存储](#)
- 使用 Ceph：[Ceph 下安装](#)

配置出多个节点

配置多个节点，与之前所描述的单机节点部署相同，使用自动化安装脚本在多个主机上部署seafile服务。安装完成后，将最先部署的单机节点上的配置文件目录

`conf` 下的所有配置文件复制替换掉刚部署的其他几个节点的配置文件。

配置后端服务器(background)

把文件搜索，office预览等后台服务从前端节点迁移到后端节点。

在复制好的某个seafile服务器上做以下配置：

安装所需的依赖库（Java,LibreOffice,poppler）

在 Ubuntu/Debian系统下：

```
sudo apt-get install openjdk-7-jre libreoffice poppler-utils python-uno # or python3-uno for ubuntu 14.04+
```

在 CentOS/Red Hat系统下：

```
sudo yum install java-1.7.0-openjdk libreoffice libreoffice-headless libreoffice-pyuno poppler-utils
```

修改后端服务器相关配置文件

编辑 **seafevents.conf** 确保以下配置信息 不存在：

```
external_es_server = true
```

编辑 **seahub_settings.py** 添加以下配置信息：

```
OFFICE_CONVERTOR_NODE = True
```

修改前端服务器相关配置文件

编辑 **seafevents.conf**, 添加以下配置信息:

```
[INDEX FILES]
external_es_server = true
es_host = <ip of node background>
es_port = 9200
```

编辑 **seahub_settings.py** 添加以下配置信息:

```
OFFICE_CONVERTOR_ROOT = 'http://<ip of node background>'
```

启动后端服务

键入以下命令以启动后台节点(注意,需要一个附加命令 **seafile-background-tasks.sh**)。

```
./seafile.sh start
./seahub.sh start-fastcgi
./seafile-background-tasks.sh start
```

关闭后端节点，键入以下命令：

```
./seafile-background-tasks.sh stop
./seafile.sh stop
./seahub.sh stop
```

设置开机自启动后端服务

在CentOS 7下添加 **/etc/systemd/system/seafile-background-tasks.service** 配置文件，添加以下内容：

```
[Unit]
Description=Seafile Background Tasks Server
After=network.target seahub.service

[Service]
Type=oneshot
ExecStart=/opt/seafile/seafile-server-latest/seafile-background-tasks.sh start
ExecStop=/opt/seafile/seafile-server-latest/seafile-background-tasks.sh stop
RemainAfterExit=yes
User=root
Group=root

[Install]
WantedBy=multi-user.target
```

执行以下命令，添加开机自启动任务：

```
systemctl enable seafile-background-tasks.service
```

license文件存放位置

使用专业版的**seafile**服务需要获取授权文件，获取该文件后应该拷贝放置到**sefile**的顶级安装目录下，使用脚本安装部署的用户请将 `license` 文件放在 `/opt/seafile` 目录下，重启服务即可生效。

负载均衡配置

HAproxy

配置示例：`/etc/haproxy/haproxy.cfg`（假设用于健康状态检测的端口为12345）

```

global
    log 127.0.0.1 local1 notice
    maxconn 4096
    user haproxy
    group haproxy

defaults
    log global
    mode http
    retries 3
    maxconn 2000
    timeout connect 10000
    timeout client 300000
    timeout server 300000

listen seafile 0.0.0.0:80
    mode http
    option httplog
    option dontlognull
    option forwardfor
    cookie SERVERID insert indirect nocache
    server seafilesserver01 <ip of frontend node1>:80 check port
12345 cookie seafilesserver01
    server seafilesserver02 <ip of frontend node2>:80 check port
12345 cookie seafilesserver02

```

启动haproxy服务，并测试使用。

修改 **SERVICE_URL** 和 **FILE_SERVER_ROOT**

下面还需要更新 **SERVICE_URL** 和 **FILE_SERVER_ROOT** 这两个配置项。否则无法通过 Web 正常的上传和下载文件。

5.0 版本开始，您可以通过管理员 Web 界面来设置这两个值 (注意，如果同时在 Web 界面和配置文件中设置了这个值，以 Web 界面的配置为准。建议在 Web 界面修改此配置。):

```

SERVICE_URL: http://<ip of haproxy node>
FILE_SERVER_ROOT: http://<ip of haproxy node>/seafhttp

```

5.0 版本之前需要修改 `ccnet.conf` 文件和 `seahub_settings.py` 文件

修改 `ccnet.conf`

```
SERVICE_URL = http://
```

修改 `seahub_settings.py`（增加一行，这是一个 `python` 文件，注意引号）

```
FILE_SERVER_ROOT = 'http://<ip of haproxy node>/seafhttp'
```

若在配置文件中修改，需要重启 `seafhttp` 和 `seahub` 服务。

高可用 `HAproxy` 节点

请参考 [高可用 HAproxy 节点](#)

`HAproxy` 下启用 `Https`

请参考 [HAproxy 下启用 Https](#)

高可用 `seafhttp` 后端节点

请参考 [高可用 seafhttp 后端节点](#)

MariaDB 集群, Memcached 集群的构建

按照[Seafile 集群文档](#)中给出的推荐架构，Seafile 集群需要使用一个分布式、高可用的数据库和缓存集群。在本文档中，我们给出一个在 3 台服务器上部署 MariaDB 和 Memcached 集群的案例。

硬件和操作系统需求

最少使用3台服务器部署来集群，每台机器都应该有：

- 2核、4GB内存。
- 1个SATA磁盘用来存储操作系统。
- 1个SATA磁盘用来存储MariaDB数据。也可以把 MariaDB 的数据保存在默认的系统目录下，这样就无需一个独立的磁盘。

具体的硬件配置可以根据管理员的经验和需求来调整。

我们使用 CentOS 7 操作系统，在下面，我们将3个服务器称为node1、node2、和node3。

部署 MariaDB 集群

MariaDB 集群的实现选择采用 MariaDB 官方推荐的集群部署方案：[MariaDB Galera Cluster](#)。MariaDB Galera Cluster 是MariaDB的一个同步多主集群架构。它只在Linux上可用，并且只支持XtraDB/InnoDB存储引擎。为了避免“脑裂”，MariaDB Galera Cluster 要求使用最少三个节点来构建集群，推荐SST(快照状态转移)使用 [rsync](#) 方式；推荐使用 [HAproxy](#) 对外部请求做负载均衡。如果你的Seafile 集群是采用[集群文档](#)中介绍的 3 节点最小化部署，则不需要用 HAProxy 来负载均衡，每个节点使用本地的 MariaDB 实例即可。

为保证集群正常工作，请务必在所有集群节点开放以下防火墙端口：

- 3306 (mysql)
- 4444 (rsync / SST)
- 4567 (galera)
- 4568 (galera IST)
- 9200 (clustercheck)

关于MariaDB Galera Cluster在CentOS下的部署详情请参考文档 [MariaDB Galera Cluster](#)

HAproxy做MariaDB集群负载均衡的配置过程请参考文档 [Setup HAProxy Load Balancer](#)

部署 memcached 集群

与其他一般的集群不同，memcached 的集群机制不是是在 memcached 服务器内部的，而是由使用 memcached 的客户端（即 Seafile 服务器）来实现 key 的分布。所以，部署 memcached 集群其实只需要在各个服务器节点上安装好 memcached 服务器程序即可。所有 Seafile 服务器，包括前端和后端服务器，都必须共享同一个 memcached 集群。

在每个节点上安装好 memcached 之后，还需要稍微修改其配置文件，以对外提供服务。

```
# 在Ubuntu系统下
vi /etc/memcached.conf

# Start with a cap of 64 megs of memory. It's reasonable, and the
# daemon default
# Note that the daemon will grow to this size, but does not start
# out holding this much
# memory
# -m 64
-m 256

# Specify which IP address to listen on. The default is to listen
# on all IP addresses
# This parameter is one of the only security measures that memcached
# has, so make sure
# it's listening on a firewalled interface.
-l 0.0.0.0

service memcached restart
```



```
# 在CentOS 7系统下
vim /etc/sysconfig/memcached

PORT="11211"
USER="memcached"
MAXCONN="1024"
CACHESIZE="64"
OPTIONS="-l 0.0.0.0 -m 256"

systemctl restart memcached
systemctl enable memcached
```

注意：为了避免重启服务后出现不必要的麻烦，请设置 memcached 服务开机自启。

后台任务节点高可用

配置**keepalived**服务

在每个seafiler后端节点上安装和配置 keepalived 来实现浮动 IP 地址。

CentOS 7:

```
yum install keepalived -y
```

假设配置了两个seafiler后台任务节点：background1、background2

在background1上修改 keepalived 配置文件(/etc/keepalived/keepalived.conf),写入如下内容：

```
! Configuration File for keepalived

global_defs {
    notification_email {
        root@localhost
    }
    notification_email_from keepalived@localhost
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id background1
    vrrp_mcast_group4 224.0.100.18
}

vrrp_instance VI_1 {
    state MASTER
    interface eno16777736
    virtual_router_id 52
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass hello123
    }
    virtual_ipaddress {
        172.26.154.43/24 dev eno16777736
    }
}
```

在background2上修改 keepalived 配置文件(/etc/keepalived/keepalived.conf),写入如下内容：

```
! Configuration File for keepalived

global_defs {
    notification_email {
        root@localhost
    }
    notification_email_from keepalived@localhost
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id background2
    vrrp_mcast_group4 224.0.100.18
}

vrrp_instance VI_1 {
    state BACKUP
    interface eno16777736
    virtual_router_id 52
    priority 98
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass hello123
    }
    virtual_ipaddress {
        172.26.154.43/24 dev eno16777736
    }
}
```

- 注意：以上配置中 `interface` 指定该节点的网卡设备名称，请根据实际情况配置。`virtual_ipaddress` 是后台服务的虚拟IP地址，也需要根据实际情况配置。

分别在两个后端节点上重启keepalived服务：

```
systemctl restart keepalived.service
```

重启成功后查看background1节点是否成功启用了相应的VIP。

修改seafile前端服务器相关配置

后台任务节点高可用

当配置好keepalived实现了虚拟IP漂移后，需要将seafile前端服务器里的相关配置指向后端服务器的虚拟IP。在各seafile前端服务器节点上：编

辑 `seafevents.conf`，修改以下配置信息：

```
[INDEX FILES]
...
es_host = <vip of nodes background>
...
```

编辑 `seahub_settings.py`，修改以下配置信息：

```
...
OFFICE_CONVERTOR_ROOT = 'http://<vip of nodes background>'
...
```

- 注意：以上配置文件中的 '`vip`' 指的是keepalived服务上配置的虚拟IP地址。

重启前端节点上的seafile、seahub服务：

```
./seafile.sh restart
./seahub.sh restart-fastcgi
```

高可用 HAproxy 节点

在每个 HAproxy 节点上安装和配置 keepalived 来实现浮动 IP 地址。

CentOS 7:

```
yum install keepalived
```

假设配置了两个 HAproxy 节点：node1、node2

在node1上修改 keepalived 配置文件(/etc/keepalived/keepalived.conf),写入如下内容：

```
! Configuration File for keepalived

global_defs {
    notification_email {
        root@localhost
    }
    notification_email_from keepalived@localhost
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id node1
    vrrp_mcast_group4 224.0.100.19
}

vrrp_instance VI_1 {
    state MASTER
    interface eno16777736
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass hello123
    }
    virtual_ipaddress {
        172.26.154.45/24 dev eno16777736
    }
}
```

在node2上修改 keepalived 配置文件(/etc/keepalived/keepalived.conf),写入如下内容：

```

! Configuration File for keepalived

global_defs {
    notification_email {
        root@localhost
    }
    notification_email_from keepalived@localhost
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id node2
    vrrp_mcast_group4 224.0.100.19
}

vrrp_instance VI_1 {
    state BACKUP
    interface eno16777736
    virtual_router_id 51
    priority 98
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass hello123
    }
    virtual_ipaddress {
        172.26.154.45/24 dev eno16777736
    }
}

```

- 注意：以上配置中 `interface` 指定该节点的网卡设备名称，请根据实际情况配置。`virtual_ipaddress` 配置HAProxy集群的虚拟IP地址，也需要根据实际情况配置。

修改 **SERVICE_URL** 和 **FILE_SERVER_ROOT**

下面还需要更新 `SERVICE_URL` 和 `FILE_SERVER_ROOT` 这两个配置项。否则无法通过 Web 正常的上传和下载文件。

5.0 版本开始，您可以通过管理员 Web 界面来设置这两个值（注意，如果同时在 Web 界面和配置文件中设置了这个值，以 Web 界面的配置为准。建议在 Web 界面修改此配置。）：


```
SERVICE_URL: http://<ip of virtual_ipaddress>  
FILE_SERVER_ROOT: http://<ip of virtual_ipaddress>/seafhttp
```

HAproxy 下启用 Hhttps

请确保您已经获取了有效的证书文件。HAproxy所需证书文件格式比较特殊，要求为pem格式，且同时包含证书和与之匹配的私钥,可使用以下命令使之合并：

```
```\n\ncat demo.crt demo.key > demo.pem\n```\n
```

## 修改 HAproxy 配置文件

配置示例：`/etc/haproxy/haproxy.cfg`（假设用于健康状态检测的端口为12345）

```

global
 log 127.0.0.1 local1 notice
 maxconn 4096
 user haproxy
 group haproxy

defaults
 log global
 mode http
 retries 3
 maxconn 2000
 timeout connect 10000
 timeout client 300000
 timeout server 300000

listen seafile
 bind :80
 bind :443 ssl crt /etc/haproxy/demo.pem
 redirect scheme https if !{ ssl_fc }
 mode http
 option httplog
 option dontlognull
 option forwardfor
 cookie SERVERID insert indirect nocache
 server seafilesver01 <ip of frontend node1>:80 check port
12345 cookie seafilesver01
 server seafilesver02 <ip of frontend node2>:80 check port
12345 cookie seafilesver02

```

## 修改 nginx 配置

在前端seafile服务器节点上（即node B 和 node C）的nginx配置中添加两行配置到

location / 代码块中： `vim /etc/nginx/conf.d/seafile.conf`

```

fastcgi_param HTTPS on;
fastcgi_param HTTP_SCHEME https;

```

配置示例：

```
location / {
 fastcgi_pass 127.0.0.1:8000;
 fastcgi_param SCRIPT_FILENAME $document_root$fastc
gi_script_name;
 fastcgi_param PATH_INFO $fastcgi_script_name
;

 fastcgi_param SERVER_PROTOCOL $server_protocol;
 fastcgi_param QUERY_STRING $query_string;
 fastcgi_param REQUEST_METHOD $request_method;
 fastcgi_param CONTENT_TYPE $content_type;
 fastcgi_param CONTENT_LENGTH $content_length;
 fastcgi_param SERVER_ADDR $server_addr;
 fastcgi_param SERVER_PORT $server_port;
 fastcgi_param SERVER_NAME $server_name;
 fastcgi_param HTTPS on;
 fastcgi_param HTTP_SCHEME https;
 ...
}
```

重新加载nginx配置：

```
nginx -s reload
```

## NFS 下集群安装

Seafile 集群中，各seafile服务器节点之间数据共享的一个常用方法是使用NFS共享存储。在NFS上共享的对象应该只是文件，这里提供了一个关于如何共享和共享什么的教程。

如何配置NFS服务器和客户端超出了本wiki的范围，提供以下参考文献：

- Ubuntu: <https://help.ubuntu.com/community/SettingUpNFSHowTo>
- CentOS: [http://www.centos.org/docs/5/html/Deployment\\_Guide-en-US/ch-nfs.html](http://www.centos.org/docs/5/html/Deployment_Guide-en-US/ch-nfs.html)

假设您使用了脚本安装,seafile的安装目录就是 `/opt/seafile` ,该目录下有一个 `seafile-data` 目录。并且，假如您挂载了NFS到 `/seafile-nfs` 目录下，请按照如下几个步骤配置：

- 将 `seafile-data` 目录移动到 `/seafile-nfs` 目录下:

```
mv /opt/seafile/seafile-data /seafile-nfs/
```

- 在集群中的每个节点上，为共享目录 `seafile-data` 设置一个软链接

```
ln -s /seafile-nfs/seafile-data /opt/seafile/seafile-data
```

这样，各seafile实例将共享同一个 `seafile-data` 目录。所有的其他配置文件和日志文件将保持独立。

# 集群升级

## 主版本和次版本升级

Seafile 在主版本和次版本中添加了新功能。有可能需要修改一些数据库表，或者需要更新搜索索引。一般来说升级集群包含以下步骤：

1. 更新数据库
2. 更新前端和后端节点上的符号链接以指向最新版本。
3. 更新每个节点上的配置文件。
4. 更新后端节点上的搜索索引。

一般来说，升级集群，您需要：

1. 在一个前端节点上运行升级脚本(例如：`./upgrade/upgrade_4_0_4_1.sh`)
2. 在其他所有节点上运行次版本升级脚本(`./upgrade/minor_upgrade.sh`)，以更新符号链接
3. 根据每个版本的文档更新每个节点上的配置文件。
4. 必要的话，在后端节点上删除旧的搜索索引。

## 维护升级

维护升级很简单，您只需要在每一个节点运行脚本

```
./upgrade/minor_upgrade.sh
```

 来更新符号链接。

## 每个版本的具体说明

### 从 5.1 到 6.0

在 6.0 版本中，文件夹下载机制已经更新，这要求在集群部署中，`seafile-data/httptemp` 文件夹必须在一个NFS共享中。您可以使该文件夹作为NFS共享的符号链接。

```
cd /data/haiwen/
ln -s /nfs-share/seafile-httptemp seafile-data/httptemp
```

httptmp文件夹仅包含用于在 Web UI 上下载/上传文件的临时文件。因此NFS共享没有任何可靠性要求，您可以从集群中的任意节点导出它。

## 从 v5.0 到 v5.1

由于 Django 升级到了 1.8，需要对 `COMPRESS_CACHE_BACKEND` 做修改：

```
- COMPRESS_CACHE_BACKEND = 'locmem://'
+ COMPRESS_CACHE_BACKEND = 'django.core.cache.backends.locmem.LocMemCache'
```

## 从 v4.4 到 v5.0

v5.0 引入了一些数据库模式更改并且所有配置文件(ccnet.conf, seafile.conf, seafevents.conf, seahub\_settings.py)被移动到了统一的配置文件目录下。

执行以下步骤做升级：

- 在一个前端节点上运行升级脚本以更新数据库。

```
./upgrade/upgrade_4.4_5.0.sh
```

- 然后，在所有的前端和后端节点上运行带有 "SEAFILE\_SKIP\_DB\_UPGRADE" 环境变量的升级脚本：

```
SEAFILE_SKIP_DB_UPGRADE=1 ./upgrade/upgrade_4.4_5.0.sh
```

升级之后，您应该查看一下配置文件已经全都被移动到 `conf/` 目录下了。

```
conf/
|__ ccnet.conf
|__ seafile.conf
|__ seafevent.conf
|__ seafdav.conf
|__ seahub_settings.conf
```

## 从 v4.3 到 v4.4

从 v4.3 到 v4.4 没有数据库和搜索索引升级。根据以下步骤做升级：

1. 在前端和后端节点上运行次版本升级脚本。

## 从 v4.2 到 v4.3

从 v4.2 到 v4.3 不包含数据库的变化，但是旧的搜索索引将被删除并重新生成。

一个新的配置项 `COMPRESS_CACHE_BACKEND = 'django.core.cache.backends.locmem.LocMemCache'` 需要被添加到 `seahub_settings.py` 中。

根据以下步骤升级：

1. 在一个前端节点上运行升级脚本来修改 `seahub_settings.py`
2. 在每一个节点上修改 `seahub_settings.py` 用新的密钥替换旧密钥，并添加配置项项 `COMPRESS_CACHE_BACKEND`
3. 在前端和后端节点上运行次版本升级脚本。
4. 删除后端节点上的旧的索引(目录 `pro_data/search`)。
5. 在后端节点上删除旧的 `office` 文件预览输出目录(`/tmp/seafile-office-output`)。



## 用户角色与权限

6.0 版本开始，管理员可以在用户管理界面为一个用户赋予一个角色，不同角色可以配置不同权限，目前支持 10 种权限。

Seafile 内建两种用户角色：`default` 和 `guest`（访客用户）。

`default` 用户实际就是一个普通 Seafile 用户，对应默认权限列表如下：

```
'default': {
 'can_add_repo': True,
 'can_add_group': True,
 'can_view_org': True,
 'can_use_global_address_book': True,
 'can_generate_share_link': True,
 'can_generate_upload_link': True,
 'can_invite_guest': False,
 'can_connect_with_android_clients': True,
 'can_connect_with_ios_clients': True,
 'can_connect_with_desktop_clients': True,
},
```

`guest` 用户对应默认权限列表如下：

```
'guest': {
 'can_add_repo': False,
 'can_add_group': False,
 'can_view_org': False,
 'can_use_global_address_book': False,
 'can_generate_share_link': False,
 'can_generate_upload_link': False,
 'can_invite_guest': False,
 'can_connect_with_android_clients': False,
 'can_connect_with_ios_clients': False,
 'can_connect_with_desktop_clients': False,
},
```

## 编辑内建用户角色的权限

通过在 `seahub_settings.py` 里加入相应配置，可以更改内建用户角色的默认权限。

比如你想让 `default` 用户可以邀请 `guest`，让 `guest` 用户可以查看公共资料库（而不改变其他权限），在 `seahub_settings.py` 里加入以下配置即可：

```
ENABLED_ROLE_PERMISSIONS = {
 'default': {
 'can_add_repo': True,
 'can_add_group': True,
 'can_view_org': True,
 'can_use_global_address_book': True,
 'can_generate_share_link': True,
 'can_generate_upload_link': True,
 'can_invite_guest': True,
 'can_connect_with_android_clients': True,
 'can_connect_with_ios_clients': True,
 'can_connect_with_desktop_clients': True,
 },
 'guest': {
 'can_add_repo': False,
 'can_add_group': False,
 'can_view_org': True,
 'can_use_global_address_book': False,
 'can_generate_share_link': False,
 'can_generate_upload_link': False,
 'can_invite_guest': False,
 'can_connect_with_android_clients': False,
 'can_connect_with_ios_clients': False,
 'can_connect_with_desktop_clients': False,
 }
}
```

注意对比 `default` 用户的 `can_invite_guest` 设置，和 `guest` 用户的 `can_view_org` 设置。

## 关于邀请访客功能

如果想要使用 邀请访客 功能，除了赋予用户 `can_invite_guest` 权限外，还需在 `seahub_settings.py` 中加入以下配置：

```
ENABLE_GUEST_INVITATION = True
```

重启 Seafile 后，可以邀请访客的用户，页面左侧会多出 邀请 导航标签。用户提供被邀请人的邮箱后，被邀请人会收到邀请链接。

## 自定义用户角色

如果你想增加一个用户角色，比如 `employee` 角色，而此角色不具有“邀请访客”、“生成上传/下载外链”权限，在 `seahub_settings.py` 中加入以下配置：

```
ENABLED_ROLE_PERMISSIONS = {
 'default': {
 'can_add_repo': True,
 'can_add_group': True,
 'can_view_org': True,
 'can_use_global_address_book': True,
 'can_generate_share_link': True,
 'can_generate_upload_link': True,
 'can_invite_guest': False,
 'can_connect_with_android_clients': True,
 'can_connect_with_ios_clients': True,
 'can_connect_with_desktop_clients': True,
 },
 'guest': {
 'can_add_repo': False,
 'can_add_group': False,
 'can_view_org': False,
 'can_use_global_address_book': False,
 'can_generate_share_link': False,
 'can_generate_upload_link': False,
 'can_invite_guest': False,
 'can_connect_with_android_clients': False,
 'can_connect_with_ios_clients': False,
 'can_connect_with_desktop_clients': False,
 },
 'employee': {
 'can_add_repo': True,
 'can_add_group': True,
 'can_view_org': True,
 'can_use_global_address_book': True,
 'can_generate_share_link': False,
 'can_generate_upload_link': False,
 'can_invite_guest': False,
 'can_connect_with_android_clients': True,
 'can_connect_with_ios_clients': True,
 'can_connect_with_desktop_clients': True,
 },
}
```

## 用户角色与权限

注意 `employee` 用户的

`can_invite_guest` 、 `can_generate_share_link` 、 `can_generate_upload_link` 设置。

## 两步认证

从6.0版本开始，增加了两步身份认证以增强账号安全。

有两种方法启用这个功能：

- 系统管理员可以在系统设置页面的“密码”部分勾选复选框，或者
- 只添加 `ENABLE_TOW_FACTOR_AUTH = True` 到 `seahub_settings.py` 并且重启服务。

之后，用户个人资料页面将会出现“两步认证”部分。

用户可以在智能手机上使用Google 身份验证器扫描二维码。

## Twilio 集成

我们也可以通过使用Twilio服务支持短信方式。

首先你需要安装 Twilio python库：

```
sudo pip install twilio
```

然后添加以下配置项到 `seahub_settings.py`：

```
TWO_FACTOR_SMS_GATEWAY = 'seahub_extra.two_factor.gateways.twilio.gateway.Twilio'
TWILIO_ACCOUNT_SID = '<your-account-sid>'
TWILIO_AUTH_TOKEN = '<your-auth-token>'
TWILIO_CALLER_ID = '<your-caller-id>'
EXTRA_MIDDLEWARE_CLASSES = (
 'seahub_extra.two_factor.gateways.twilio.middleware.ThreadLocals',
)
```

注意：如果你已经定义了 `EXTRA_MIDDLEWARE_CLASSES` ,请使用

```
EXTRA_MIDDLEWARE_CLASSES += (替换掉 EXTRA_MIDDLEWARE_CLASSES = (
```

然后重启，当用户为其帐户启用两步认证时，将会显示“短信”方法。



## 向seafile中导入目录

从seafile专业版5.1.3开始，支持将服务器上的一个本地文件目录导入到seafile中。它是系统管理员从现有的文件服务器(NFS,Samba etc.)中导入文件的便利工具。

要导入一个目录，应该在 seafile-server-laster 目录下使用 `seaf-import.sh` 脚本。

```
usage :
seaf-import.sh
 -p <import dir path, must set>
 -n <repo name, must set>
 -u <repo owner, must set>
```

指定的目录将作为一个资料库被导入到seafile中。你可以设定被导入资料库的名字和所有者。

执行 `./seaf-import.sh -p <dir you want to import> -n <repo name> -u <repo owner>` ,

```
Starting seaf-import, please wait ...
[04/26/16 03:36:23] seaf-import.c(79): Import file ./runtime/sea
hub.pid successfully.
[04/26/16 03:36:23] seaf-import.c(79): Import file ./runtime/err
or.log successfully.
[04/26/16 03:36:23] seaf-import.c(79): Import file ./runtime/sea
hub.conf successfully.
[04/26/16 03:36:23] seaf-import.c(79): Import file ./runtime/acc
ess.log successfully.
[04/26/16 03:36:23] seaf-import.c(183): Import dir ./runtime/ to
repo 5ffb1f43 successfully.
run done
Done.
```

使用指定的资料库所有者登陆到seafile服务，你将会发现一个指定名称的新资料库。



## 配置选项

**Note:** Seafile 服务器 5.0.0 之后，所有配置文件都移动到了统一的 **conf** 目录下。  
[了解详情](#).

在 `/data/haiwen/conf/seafevents.conf` 配置文件中：

```
[AUDIT]
审计日志默认是关闭的
enabled = true

[INDEX FILES]
要启用搜索，必须设置为 "true"
enabled = true

搜索索引更新的时间间隔。可以是 s（秒）， m（分）， h（小时）， d（天）
interval=10m

如果设置为 "true"，搜索索引更新时也会索引办公文件和 pdf 文件中的内容
注意： 如果您将此选项从 "false" 设置为 "true"， 那么您需要清空搜索索引
然后再次更新索引。更多详情请参考 FAQ。
index_office_pdf=false

[OFFICE CONVERTER]

要启用办公文件和 pdf 文件的在线预览功能必须设置为 "true"
enabled = true

能够并发运行的 libreoffice 工作进程数
workers = 1

转换后的办公文件和 pdf 文件的存储位置
outputdir = /tmp/

允许的最大在线预览页数。默认为 50 页
max-pages = 50

允许的最大预览文件的大小，单位是 MB。默认为 2 MB
max-size = 2
```

```
[SEAHUB EMAIL]
```

```
要启用用户邮件通知，必须设置为 "true"
```

```
enabled = true
```

```
发送 seahub 邮件的时间间隔。可以是 s (秒)， m (分)， h (小时)， d (天)
```

```
interval = 30m
```

## 您可能想要更改的配置选项

以上小节已经列出了 `/data/haiwen/conf/seafevents.conf` 配置文件中的所有配置选项。大多数情况下，使用默认配置就足够了。但是为了更好地满足自身需求，您可能想要更改其中的某些选项。

我们将这些配置选项列出在下面的表中，以及我们选择默认设置的原因。

| 段                | 选项               | 默认值   | 描述                                                                                                                                      |
|------------------|------------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------|
| INDEX FILES      | index_office_pdf | false | 默认情况下，office 文档和 pdf 文档的全文搜索功能是不开启的。这是因为它会占用相当大的搜索索引空间。要开启它，将它的值设置为 "true" 然后重新创建搜索索引。更多详情请参考 <a href="#">[[Seafile 专业版服务器的 FAQ]]</a> 。 |
| OFFICE CONVERTER | max-size         | 2MB   | 允许的最大在线预览文件的大小是 2MB。在线预览会把 office 和 pdf 文档转换成 HTML 然后在浏览器中显示。如果文件太大，转换会花费很长时间且占用很多空间。                                                   |
| OFFICE CONVERTER | max-pages        | 50    | 当在线预览一个 office 或者 pdf 文档时，文档的前 50 页将会首先被显示。如果此值太大，转换会花费很长时间且占用很多空间。                                                                     |

- [关于搜索功能的 FAQ](#)
- [Office/PDF 文档预览 FAQ](#)

## 关于搜索功能的 FAQ

- 无论我怎样尝试，加密资料库中的文件都不会出现在搜索结果中

这是因为服务器不能索引加密文件，因为它们是加密的。

- 我从社区版服务器切换到专业版服务器后，无论我搜索什么，都不会得到结果

默认情况下，搜索索引每 10 分钟更新一次。所以，在第一次索引更新前，无论您搜索什么都不会返回结果。

为了使搜索能够立即生效，您可以手动更新索引：

- 确保您已经启动了 Seafile 服务器
- 手动更新搜索索引：

```
cd haiwen/seafile-pro-server-2.1.5
./pro/pro.py search --update
```

如果您的文件很多，这个过程会花费很长一段时间。

- 我想启用对 office/pdf 文档的全文搜索功能，所以我在配置文件中将 `index_office_pdf` 的值设置为 `true`，但它没起作用。

在这种情况下，您需要做以下几步：

1. 编辑 `/data/haiwen/conf/seafevents.conf` 文件，将 `index_office_pdf` 的值设置为 `true`
2. 重启 Seafile 服务器：

```
cd /data/haiwen/seafile-pro-server-2.1.5
./seafile.sh restart
```

3. 删除已经存在的搜索索引：

```
./pro/pro.py search --clear
```

4. 创建并且再次更新搜索索引：

```
./pro/pro.py search --update
```

## Office/PDF 文档预览 FAQ

### 怎么修改可预览最大文件大小和页面数？

在 `/data/haiwen/conf/seafevents.conf` 中的 `OFFICE CONVERTER` 配置部分添加配置选项

```
the max size of documents to allow to be previewed online, in
MB. Default is 2 MB
max-size = 2
how many pages are allowed to be previewed online. Default is
50 pages
max-pages = 50
```

然后重启 Seafile 服务

```
cd /data/haiwen/seafile-pro-server-1.7.0/
./seafile.sh restart
./seahub.sh restart
```

Seafile Professional Edition  
SOFTWARE LICENSE AGREEMENT

NOTICE: READ THE FOLLOWING TERMS AND CONDITIONS CAREFULLY BEFORE YOU DOWNLOAD, INSTALL OR USE Seafile, Inc'S PROPRIETARY SOFTWARE. BY INSTALLING OR USING THE SOFTWARE, YOU AGREE TO BE BOUND BY THE FOLLOWING TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THE FOLLOWING TERMS AND CONDITIONS, DO NOT INSTALL OR USE THE SOFTWARE.

1. DEFINITIONS

"Seafile, Inc" means Seafile, Inc

"You and Your" means the party licensing the Software hereunder.

"Software" means the computer programs provided under the terms of this license by Seafile, Inc together with any documentation provided therewith.

2. GRANT OF RIGHTS

2.1 General

The License granted for Software under this Agreement authorizes You on a non-exclusive basis to use the Software. The Software is licensed, not sold to You and Seafile, Inc reserves all rights not expressly granted to You in this Agreement. The License is personal to You and may not be assigned by You to any third party.

2.2 License Provisions

Subject to the receipt by Seafile, Inc of the applicable license fees, You have the right use the Software as follows:

- \* You may use and install the Software on an unlimited number of computers that are owned, leased, or controlled by you.
- \* Nothing in this Agreement shall permit you, or any third party to disclose or otherwise make available to any third party the licensed Software, source code or any portion thereof.
- \* You agree to indemnify, hold harmless and defend Seafile, Inc from and against any claims or lawsuits, including attorney's fees, that arise as a result from the use of the Software;
- \* You do not permit further redistribution of the Software by Your end-user customers

## 5. NO DERIVATIVE WORKS

The inclusion of source code with the License is explicitly not for your use to customize a solution or re-use in your own projects or products. The benefit of including the source code is for purposes of security auditing. You may modify the code only for emergency bug fixes that impact security or performance and only for use within your enterprise. You may not create or distribute derivative works based on the Software or any part thereof. If you need enhancements to the software features, you should suggest them to Source Tree Solutions for version improvements.

## 6. OWNERSHIP

You acknowledge that all copies of the Software in any form are the sole property of Seafile, Inc. You have no right, title or interest to any such Software or copies thereof except as provided in this Agreement.

## 7. CONFIDENTIALITY

You hereby acknowledge and agreed that the Software constitute and contain valuable proprietary products and trade secrets of Seafile, Inc, embodying substantial creative efforts and confidential information, ideas, and expressions. You agree to treat, and take precautions to ensure that your employees and other third parties treat, the Software as confidential in accordance with the confidentiality requirements herein.

## 8. DISCLAIMER OF WARRANTIES

EXCEPT AS OTHERWISE SET FORTH IN THIS AGREEMENT THE SOFTWARE IS PROVIDED TO YOU "AS IS", AND Seafile, Inc MAKES NO EXPRESS OR IMPLIED WARRANTIES WITH RESPECT TO ITS FUNCTIONALITY, CONDITION, PERFORMANCE, OPERABILITY OR USE. WITHOUT LIMITING THE FOREGOING, Seafile, Inc DISCLAIMS ALL IMPLIED WARRANTIES INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR FREEDOM FROM INFRINGEMENT. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSIONS MAY NOT APPLY TO YOU. THE LIMITED WARRANTY HEREIN GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS THAT VARY FROM ONE JURISDICTION TO ANOTHER.

## 9. LIMITATION OF LIABILITY

YOU ACKNOWLEDGE AND AGREE THAT THE CONSIDERATION WHICH Seafile,

Inc IS CHARGING HEREUNDER DOES NOT INCLUDE ANY CONSIDERATION FOR ASSUMPTION BY Seafire, Inc OF THE RISK OF YOUR CONSEQUENTIAL OR INCIDENTAL DAMAGES WHICH MAY ARISE IN CONNECTION WITH YOUR USE OF THE SOFTWARE. ACCORDINGLY, YOU AGREE THAT Seafire, Inc SHALL NOT BE RESPONSIBLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS-OF-PROFIT, LOST SAVINGS, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF A LICENSING OR USE OF THE SOFTWARE.

#### 10. INDEMNIFICATION

You agree to defend, indemnify and hold Seafire, Inc and its employees, agents, representatives and assigns harmless from and against any claims, proceedings, damages, injuries, liabilities, costs, attorney's fees relating to or arising out of Your use of the Software or any breach of this Agreement.

#### 11. TERMINATION

Your license is effective until terminated. You may terminate it at any time by destroying the Software or returning all copies of the Software to Seafire, Inc. Your license will terminate immediately without notice if You breach any of the terms and conditions of this Agreement, including non or incomplete payment of the license fee. Upon termination of this Agreement for any reason: You will uninstall all copies of the Software; You will immediately cease and desist all use of the Software; and will destroy all copies of the software in your possession.

#### 12. UPDATES AND SUPPORT

Seafire, Inc has the right, but no obligation, to periodically update the Software, at its complete discretion, without the consent or obligation to You or any licensee or user.

#### 13. TAXES AND OTHER CHARGES

You are responsible for paying all sales, use, excise, valuated or other taxes or governmental charges in addition to freight, insurance and installation charges and import or export duties.

YOU HEREBY ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS.

## 服务器配置

### 配置文件

注意: Seafile 服务器 5.0.0 之后, 所有配置文件都移动到了统一的 **conf** 目录下, 而且大部分配置选项可以通过 Web 界面 (Admin Panel -> Settings) 来配置, 不需要直接修改配置文件。通过 Web 界面做的配置会保存在数据库中, 并且优先级比通过配置文件做的配置的优先级高。

开源版中包括以下三个配置文件:

- [conf/ccnet.conf](#): 用来配置网络和 LDAP/AD 连接
- [conf/seafile.conf](#): 用来配置 Seafile
- [conf/seahub\\_settings.py](#): 用来配置 Seahub

专业版中还包含以下一个配置文件:

- `conf/seafevents.conf` : 包含搜索与文件预览的配置

### 配置项

邮件:

- [发送邮件提醒](#)
- [个性化邮件提醒](#)

用户管理:

- [用户管理](#)

用户存储容量和上传/下载文件大小限制:

- [存储容量与文件上传/下载大小限制](#)

### 自定义 Web

- [自定义 Web](#)



## ccnet.conf 配置

**Note:** Seafile 服务器 5.0.0 之后，所有配置文件都移动到了统一的 **conf** 目录下。  
[了解详情](#).

```
[General]
该设置不再使用
USER_NAME=example

请不要改变这个 ID.
ID=eb812fd276432eff33bcdde7506f896eb4769da0

该设置不再使用
NAME=example

Seahub (Seafile Web) 外部 URL，如果该值没有设对，会影响文件的上传下载
注意：外部 URL 意味着"如果你使用 Nginx，请使用 Nginx 对外的 URL"
5.0 版开始，建议通过 Web 界面来修改，不要直接修改 ccnet.conf 中的值
SERVICE_URL=http://www.example.com:8000

[Network]
该设置不再使用
PORT=10001

[Client]
该设置不再使用
PORT=13419
```

## 注意

为使更改生效，请重启 Seafile

```
./seafile.sh restart
```

## seafile.conf 配置

**Note:** Seafile 服务器 5.0.0 之后，所有配置文件都移动到了统一的 **conf** 目录下。  
[了解详情](#).

### 存储空间容量设置

用户默认空间上限

```
[quota]
单位为 GB
default = 2
```

这个设置对所有用户生效. 如果你想对某一特定用户进行容量分配, 请以管理员身份登陆 Seahub 网站, 在 **System Admin** 页面中进行设置.

### 默认历史记录设置

对所有的资料库设置一个默认的文件历史保留天数：

```
[history]
keep_days = days of history to keep
```

## Seafile fileserver

Seafile 监听的端口号 (不要修改该设置)

```
[fileserver]
Seafile tcp 端口 (不要修改该设置)
port = 8082
```

上传/下载大小限制：

```
[fileserver]
上传文件最大为200M.
max_upload_size=200

最大下载目录限制为200M.
max_download_dir_size=200
```

## 注意

请重启 Seafile 和 Seahub 以使修改生效：

```
./seahub.sh restart
./seafile.sh restart
```

## Seahub 配置

**Note:** Seafile 服务器 5.0.0 之后，所有配置文件都移动到了统一的 **conf** 目录下。  
[了解详情](#).

## Seahub 下发送邮件提醒

请参看 [发送邮件提醒](#)

## 缓存

Seahub 在默认文件系统(/tmp/seahub\_cache/)中缓存文件(avatars, profiles, etc) .  
 你可以通过 Memcached 进行缓存操作 (前提是你已经安装了 `python-memcache` 模块).

```
CACHES = {
 'default': {
 'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
 'LOCATION': '127.0.0.1:11211',
 }
}
```

## 用户管理选项

The following options affect user registration, password and session.

```
是非开启用户注册功能。默认为 `False`。
ENABLE_SIGNUP = False

用户注册后是否立刻激活，默认为 `True`。
如设置为 `False`，需管理员手动激活。
ACTIVATE_AFTER_REGISTRATION = False

管理员新增用户后是否给用户发送邮件。默认为 `True`。
SEND_EMAIL_ON_ADDING_SYSTEM_MEMBER = True
```

```
管理员重置用户密码后是否给用户发送邮件。默认为 `True`。
SEND_EMAIL_ON_RESETTING_USER_PASSWD = True

登录记住天数。默认 7 天
LOGIN_REMEMBER_DAYS = 7

用户输入密码错误次数超过改设置后，显示验证码
LOGIN_ATTEMPT_LIMIT = 3

如果登录密码输错次数超过 ``LOGIN_ATTEMPT_LIMIT``，冻结账号
since 5.1.2
FREEZE_USER_ON_LOGIN_FAILED = True

用户密码最少长度
USER_PASSWORD_MIN_LENGTH = 6

检查用户密码的复杂性
USER_STRONG_PASSWORD_REQUIRED = False

用户密码复杂性：
数字，大写字母，小写字母，其他符号
'3' 表示至少包含以上四种类型中的 3 个
USER_PASSWORD_STRENGTH_LEVEL = 3

管理员添加／重置用户后，强制用户修改登录密码
在版本 5.1.1 加入，默认开启
FORCE_PASSWORD_CHANGE = True

cookie 的保存时限，(默认为 2 周)。
SESSION_COOKIE_AGE = 60 * 60 * 24 * 7 * 2

浏览器关闭后，是否清空用户会话 cookie
SESSION_EXPIRE_AT_BROWSER_CLOSE = False

是否存储每次请求的会话数据。默认为 `False`
SESSION_SAVE_EVERY_REQUEST = False
```

## 资料库设置

```
加密资料库密码最小长度
REPO_PASSWORD_MIN_LENGTH = 8

加密外链密码最小长度
SHARE_LINK_PASSWORD_MIN_LENGTH = 8

关闭与任意目录同步的功能
DISABLE_SYNC_WITH_ANY_FOLDER = True

允许用户设置资料库的历史保留天数
ENABLE_REPO_HISTORY_SETTING = True
```

## 在线文件查看设置

```
是否使用 pdf.js 来在线查看文件。默认为 `True`
USE_PDFJS = True

在线文件查看最大文件大小，默认为 30M。
注意，在专业版中，seafevents.conf 中有另一个选项
`max-size` 也控制 doc/ppt/excel/pdf 文件在线查看的最大文件大小。
您需要同时把这两个选项调大，如果您要允许 30M 以上 doc/ppt/excel/pdf 的查看。
FILE_PREVIEW_MAX_SIZE = 30 * 1024 * 1024

开启 thumbnails 功能
ENABLE_THUMBNAIL = True

文件缩略图的存储位置
THUMBNAIL_ROOT = '/haiwen/seahub-data/thumbnail/thumb/'

开启或禁用视频缩略图
NOTE: since version 6.1
ENABLE_VIDEO_THUMBNAIL = False

使用第5秒的图片作为缩略图
THUMBNAIL_VIDEO_FRAME_TIME = 5
```

## 其他选项

```
时区设置
可用的时区参考:
http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
TIME_ZONE = 'UTC'

系统默认语言设置
可用的设置值参考
http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = 'en'

站点名, 用在 Email 中.
SITE_NAME = 'example.com'

站点 Title
SITE_TITLE = 'Seafile'
```

## 专业版选项

```
是否允许管理员通过 Web 查看用户非加密资料库. 默认为 False
ENABLE_SYS_ADMIN_VIEW_REPO = True

未登录用户, 外链页面下载/上传需要提供邮箱, 做审计。
Since version 5.1.4
ENABLE_SHARE_LINK_AUDIT = True
```

## RESTful API

```
API throttling 相关配置。如果api的返回码为429，可以调高下面的数值。
REST_FRAMEWORK = {
 'DEFAULT_THROTTLE_RATES': {
 'ping': '600/minute',
 'anon': '5/minute',
 'user': '300/minute',
 },
 'UNICODE_JSON': False,
}

Throtting 白名单，用来忽略特定IP。
e.g. REST_FRAMEWORK_THROTTING_WHITELIST = ['127.0.0.1', '192.168.1.1']
请确保 `REMOTE_ADDR` 头部在 Nginx 配置了，具体参考 https://manual.seafhub.com/deploy/deploy_with_nginx.html
REST_FRAMEWORK_THROTTING_WHITELIST = []
```

## 注意

- 请重启 Seahub 以使更改生效。
- 如果更改没有生效，请删除 `seahub_setting.pyc` 缓存文件。
- 如果需要在 `seahub_settings.py` 里添加中文注释，请把 `# -*- coding: utf-8 -*-` 写入文件第一行，并单独为一行。



## 发送邮件提醒

邮件提醒会使某些功能有更好的用户体验, 比如发送邮件提醒用户新消息到达. 请在 `seahub_settings.py` 中加入以下语句以开启邮件提醒功能 (同时需要对你的邮箱进行设置).

```
EMAIL_USE_TLS = False
EMAIL_HOST = 'smtp.domain.com' # smtp 服务器
EMAIL_HOST_USER = 'username@domain.com' # 用户名和域名
EMAIL_HOST_PASSWORD = 'password' # 密码
EMAIL_PORT = '25'
DEFAULT_FROM_EMAIL = EMAIL_HOST_USER
SERVER_EMAIL = EMAIL_HOST_USER
```

Gmail 邮箱示例:

```
EMAIL_USE_TLS = True
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_HOST_USER = 'username@gmail.com'
EMAIL_HOST_PASSWORD = 'password'
EMAIL_PORT = 587
DEFAULT_FROM_EMAIL = EMAIL_HOST_USER
SERVER_EMAIL = EMAIL_HOST_USER
```

QQ 邮箱示例 (只能在没有使用 TLS 的情况下配置成功, 不安全):

```
EMAIL_USE_TLS = False
EMAIL_HOST = 'smtp.exmail.qq.com'
EMAIL_HOST_USER = 'username@domain.com'
EMAIL_HOST_PASSWORD = 'password'
EMAIL_PORT = '25'
DEFAULT_FROM_EMAIL = EMAIL_HOST_USER
SERVER_EMAIL = EMAIL_HOST_USER
```

163 邮箱未测试成功 (有些国内公共邮箱做了限制的, 是不能配置成功的.)

126 邮箱:

```
EMAIL_USE_TLS = True
EMAIL_HOST = 'smtp.vip.126.com'
EMAIL_HOST_USER = 'test@vip.126.com'
EMAIL_HOST_PASSWORD = 'password'
EMAIL_PORT = 25
DEFAULT_FROM_EMAIL = EMAIL_HOST_USER
SERVER_EMAIL = EMAIL_HOST_USER
```

注意1:如果邮件功能不能正常使用,请在 `logs/seahub.log` 日志文件中查看问题原因. 更多信息请见 [Email notification list](#).

推荐以下调试方法:

- 在管理员界面添加一个用户
- 如果界面上报告邮件发送出错,检查下 `logs/seahub.log`
- 如果日志中有这样的错误 `seahub.views.sysadmin:1334 user_add [Errno 111] Connection refused`, 那么是邮件服务器地址或端口号配置有问题。可以参考 <http://stackoverflow.com/questions/5802189/django-errno-111-connection-refused>

注意2: 如果你想在非用户验证情况下使用邮件服务, 请将 `EMAIL_HOST_USER` 和 `EMAIL_HOST_PASSWORD` 置为 **blank** ( `' '` ). (但是注意一点, 这种情况下, 邮件将不会记录发件人 `From:` 信息.)

注意3:

- 请重启 Seahub 以使更改生效.
- 如果更改没有生效, 请删除 `seahub_setting.pyc` 缓存文件.

```
./seahub.sh restart
```

## 个性化邮件提醒

注意: 不同版本之间有所差异, 本文档基于 2.0.1 版本编写。请按提示, 自行更改相应代码, 以实现个性化功能。重启 Seahub 以使更改生效。

### 用户重置密码

#### Subject

seahub/seahub/auth/forms.py line:103

#### Body

seahub/seahub/templates/registration/password\_reset\_email.html

### 管理员添加新用户

#### Subject

seahub/seahub/views/sysadmin.py line:424

#### Body

seahub/seahub/templates/sysadmin/user\_add\_email.html

### 管理员重置用户密码

#### Subject

seahub/seahub/views/sysadmin.py line:368

#### Body

seahub/seahub/templates/sysadmin/user\_reset\_email.html

### 用户发送文件/文件夹外链

#### Subject

seahub/seahub/share/views.py line:668

## Body

seahub/seahub/templates/shared\_link\_email.html

## 存储容量与文件上传/下载大小限制

**Note:** Seafile 服务器 5.0.0 之后，所有配置文件都移动到了统一的 **conf** 目录下。  
[了解详情](#).

### 存储容量

可以通过在 `seafile.conf` 文件中增加以下语句，来为所有用户设置默认存储容量（比如，2GB）。

```
[quota]
用户存储容量，单位默认为 GB，要求为整数。
default = 2
```

在社区版5.0.5以后，你可以以 KB, MB, GB, TB 为单位来设置默认容量。比如

```
[quota]
default = 200MB
```

注意这里 1TB = 1000GB = 1000\*1000MB 以此类推。

此设置对所有用户有效，如果想为某一用户单独设置，请在管理员界面更改。

### 文件修改历史保存期限 (**seafile.conf**)

如果你不想保存所有的文件修改历史，可以在 `seafile.conf` 中设置:

```
[history]
文件修改历史保存期限（单位为“天”）
keep_days = 10
```

### 文件上传/下载大小限制

在 `seafile.conf` 中:

```
[fileserv
```

```
设置最大上传文件为 200M.
```

```
max_upload_size=200
```

```
设置最大下载文件/目录为 200M.
```

```
max_download_dir_size=200
```

## 个性化 Seahub

### 个性化 Logo 及 CSS 样式

创建 `<seafile-install-path>/seahub-data/custom` 目录. 在 `seafile-server-latest/seahub/media` 目录下创建一个符号链接: `ln -s ../../../../seahub-data/custom custom`.

升级过程中, Seafile 升级脚本会自动创建符号链接以维持个性化设置

### 个性化 Logo

1. 将 Logo 文件放在 `seafile-server-latest/seahub/media/custom/` 文件夹下
2. 在 `seahub_settings.py` 中, 重新定义 `LOGO_PATH` 的值。

```
LOGO_PATH = 'custom/mylogo.png'
```

3. 在 `seahub_settings.py` 中, 重新定义 Logo 宽高的值。

```
LOGO_WIDTH = 149
LOGO_HEIGHT = 32
```

### 个性化 Favicon

1. 将 favicon 文件放在 `seafile-server-latest/seahub/media/custom/` 文件夹下
2. 在 `seahub_settings.py` 中, 重新定义 `FAVICON_PATH` 的值。

```
FAVICON_PATH = 'custom/favicon.png'
```

### 自定义 Seahub CSS 样式

1. 在 `seahub/media/custom/` 中新建 CSS 文件, 比如: `custom.css`。
2. 在 `seahub_settings.py` 中, 重新定义 `BRANDING_CSS` 的值。

```
BRANDING_CSS = 'custom/custom.css'
```

## 个性化 Seahub 页面

在 `<seafile-install-path>/seahub-data/custom` 目录下，新建 `templates` 文件夹。

## 个性化“页脚”页面

注意: 6.0 版本之后，Seafile web 页面使用全屏设计, 不再使用页脚。

1. 复制 `seahub/seahub/templates/footer.html` 到 `seahub-data/custom/templates` 。
2. 自行编写 `footer.html` 。

## 个性化“下载”页面

1. 复制 `seahub/seahub/templates/download.html` 到 `seahub-data/custom/templates` 。
2. 自行编写 `download.html` 。

## 个性化“帮助”页面

1. 复制 `seahub/seahub/help/templates` 目录到 `seahub-data/custom/` 。
2. 自行编写 `seahub-data/custom/templates/help` 目录下的 html 文件。



## 管理员

### 进入管理界面

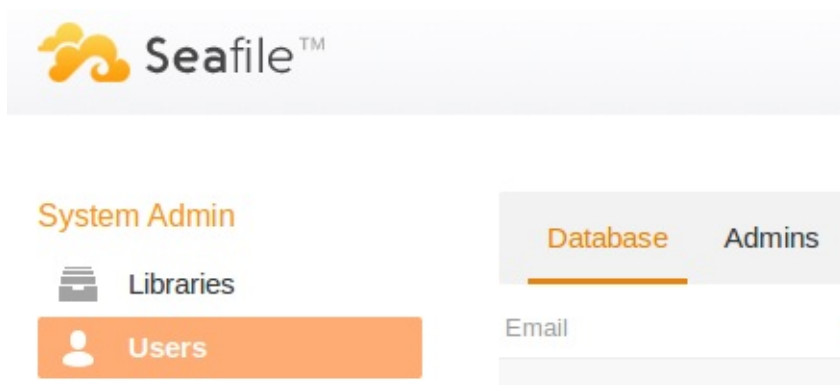
作为系统管理员，你可以通过点击网页右上侧的用户头像，在下拉菜单中点击 **System Admin** 按钮进入管理界面：



如果你使用的 Seafile 版本低于 6.0.0，可以通过点击网页右上侧的 **tools** 按钮（在个人头像旁）进入管理界面：



点击完 **tools** 按钮，你便进入管理界面：



### 账号管理

日常维护

- [账号管理](#)

## 日志

- [日志文件位置](#)

## 备份和恢复

备份和恢复:

- [备份和恢复](#)

服务器强制关闭或系统坏掉后，恢复损坏的文件：

- [Seafile FSCK](#)

你可以运行Seafile GC来删除无用的文件:

- [Seafile GC](#)

## 账号管理

### 用户管理

当你部署好 Seahub 网站，你应该已经创建好一个管理员账号。用管理员账号登陆后，便可添加、删除，用户、资料库等。

### 重置用户密码

在 **System Admin** 页面，管理员可以重置用户密码。

对于私有服务器，默认设置不支持用户通过邮箱来重置密码。如果你想采用这种方式，你必须首先[设置邮件通知](#)。

### 如果忘记管理员账号或密码如何处理？

你可以进入 `seafile-server` 目录，运行 `reset-admin.sh` 脚本。这个脚本可以帮助你重置管理员账号和密码。

## 日志

Seafile 服务器有如下日志文件:

- Ccnet Log: logs/ccnet.log
- Seafile server : logs/seafile.log
- FileServer: logs/http.log
- Controller: logs/controller.log
- Seahub : logs/seahub\_django\_request.log, logs/seahub.log

## 清理数据库

### Seahub

#### 清理 Session 数据库

清理 Session 表:

```
cd <install-path>/seafile-server-latest
./seahub.sh clearsessions
```

#### 文件活动 (Activity)

要清理文件活动表，登录到 MySQL/MariaDB，然后使用以下命令:

```
use seahub_db;
DELETE FROM Event WHERE to_days(now()) - to_days(timestamp) > 90
;
```

## 概述

一般来说，Seafile 备份分为两部分内容：

- Seafile 资料库数据
- 数据库

如果你根据我们的手册来安装 Seafile 服务器，你应该有如下目录结构：

```
haiwen # 根目录，haiwen 为示例文件名，如果你安装到其他目录则为相应的目录名
--seafile-server-2.x.x # Seafile 安装包解压缩后目录
--seafile-data # Seafile 配置文件和数据（如果你选择默认方式）
--seahub-data # Seahub 数据
--ccnet # Ccnet 配置文件和数据
--seahub.db # Seahub 用到的 sqlite3 数据库文件
--seahub_settings.py # seahub可选属性配置文件
```

你所有的资料库数据都存储在 **haiwen** 目录。

Seafile 也在数据库中存储一些重要的元数据。数据库的命名和存储路径取决于你所使用的数据库。

对于 SQLite, 数据库文件也存储在 **haiwen** 目录。相应的数据文件如下：

- ccnet/PeerMgr/usermgr.db: 包含用户信息
- ccnet/GroupMgr/groupmgr.db: 包含群组信息
- seafile-data/seafile.db: 包含资料库元数据信息
- seahub.db: 包含网站前端（Seahub）所用到的数据库表信息

对于 MySQL, 数据库由管理员来创建，所以不同的人部署，可能会有不同的文件名。大体而言，有如下三个数据库会被创建：

- ccnet-db: 包含用户和群组信息
- seafile-db: 包含资料库元数据信息
- seahub.db: 包含网站前端（seahub）所用到的数据库表信息

## 备份步骤

备份需要如下三步：

1. 可选步: 如果你选择 SQLite 作为数据库, 首先停掉 Seafile 服务器;
2. 备份数据库;
3. 备份存放 Seafile 数据的目录;

我们假设你的 Seafile 数据位于 `/data/haiwen` 目录下, 并且你想将其备份到 `/backup` 目录 (`/backup` 目录可以是 NFS (网络文件系统), 可以是另一台机器的 Windows 共享, 或者是外部磁盘)。请在 `/backup` 目录下创建如下目录结构:

```
/backup
---- databases/ 包含数据库备份
---- data/ 包含 Seafile 数据备份
```

## 备份数据库

我们建议你每次将数据库备份到另一个单独文件, 并且不要覆盖最近一周来备份过的旧数据库文件。

### MySQL

假设你的数据库名分别为 `ccnet-db`, `seafile-db` 和 `seahub-db`。 `mysqldump` 会自动锁住表, 所以在你备份 MySQL 数据库的时候, 不需要停掉 Seafile 服务器。通常因为数据库表非常小, 所以执行以下命令备份不会花太长时间。

```
mysqldump -h [mysqlhost] -u[username] -p[password] --opt ccnet-db > /backup/databases/ccnet-db.sql.`date +%Y-%m-%d-%H-%M-%S`

mysqldump -h [mysqlhost] -u[username] -p[password] --opt seafile-db > /backup/databases/seafile-db.sql.`date +%Y-%m-%d-%H-%M-%S`

mysqldump -h [mysqlhost] -u[username] -p[password] --opt seahub-db > /backup/databases/seahub-db.sql.`date +%Y-%m-%d-%H-%M-%S`
```

### SQLite

对于 SQLite 数据库, 在备份前你需要停掉 Seafile 服务器。

```
sqlite3 /data/haiwen/ccnet/GroupMgr/groupmgr.db .dump > /backup/
databases/groupmgr.db.bak.`date +%Y-%m-%d-%H-%M-%S`\`

sqlite3 /data/haiwen/ccnet/PeerMgr/usermgr.db .dump > /backup/da
tabases/usermgr.db.bak.`date +%Y-%m-%d-%H-%M-%S`\`

sqlite3 /data/haiwen/seafile-data/seafile.db .dump > /backup/dat
abases/seafile.db.bak.`date +%Y-%m-%d-%H-%M-%S`\`

sqlite3 /data/haiwen/seahub.db .dump > /backup/databases/seahub.
db.bak.`date +%Y-%m-%d-%H-%M-%S`\`
```

## 备份 Seafile 资料库数据

由于所有的数据文件都存储在 `/data/haiwen` 目录, 备份整个目录即可。你可以直接拷贝整个目录到备份目录, 或者你也可以用 `rsync` 做增量备份。

直接拷贝整个数据目录,

```
cp -R /data/haiwen /backup/data/haiwen-`date +%Y-%m-%d-%H-%M-%S`
``
```

这样每次都会产生一个新的备份文件夹, 完成后, 可以删掉旧的备份。

如果你有很多数据, 拷贝整个数据目录会花很多时间, 这时你可以用 `rsync` 做增量备份。

```
rsync -az /data/haiwen /backup/data
```

这个命令数据备份到 `/backup/data/haiwen` 下。

让拷贝和 `rsync` 过程成功结束是非常重要的, 否则你最近的一些数据将会丢失。

## 恢复备份

如果你当前的 **Seafile** 服务器已经坏掉, 将使用另一台机器来提供服务, 需要恢复数据:

1. 假设在新机器中, **Seafile** 也被部署在了 `/data/haiwen` 目录中, 拷贝



`/backup/data/haiwen` 到新机器中即可。

## 2. 恢复数据库。

# 恢复数据库

现在你已经拥有了数据库备份文件，你可以按如下步骤来进行恢复。

## MySQL

```
mysql -u[username] -p[password] ccnet-db < ccnet-db.sql.2013-10-19-16-00-05
mysql -u[username] -p[password] seafile-db < seafile-db.sql.2013-10-19-16-00-20
mysql -u[username] -p[password] seahub-db < seahub-db.sql.2013-10-19-16-01-05
```

## SQLite

```
cd /data/haiwen
mv ccnet/PeerMgr/usermgr.db ccnet/PeerMgr/usermgr.db.old
mv ccnet/GroupMgr/groupmgr.db ccnet/GroupMgr/groupmgr.db.old
mv seafile-data/seafile.db seafile-data/seafile.db.old
mv seahub.db seahub.db.old
sqlite3 ccnet/PeerMgr/usermgr.db < usermgr.db.bak.xxxx
sqlite3 ccnet/GroupMgr/groupmgr.db < groupmgr.db.bak.xxxx
sqlite3 seafile-data/seafile.db < seafile.db.bak.xxxx
sqlite3 seahub.db < seahub.db.bak.xxxx
```

# Seafile FSCK

在服务器端，Seafile 通过一种内部格式将文件存储在资料库中。Seafile 对于文件和目录有其独有的保存方式（类似于Git）。

默认安装下，这些内部对象，会被直接存储在服务器的文件系统中（例如 Ext4，NTFS）。由于大多数文件系统，不能在服务器非正常关闭或系统崩溃后，保证文件内容的完整性。所以，如果当系统崩溃时，正在有新的内部对象被写入，那么当系统重启时，这些文件就会被损坏，相应的资料库也无法使用。

注意: 如果你把 seafdata 目录存储在有后备电源的 NAS（例如 EMC 或 NetApp）系统中，或者使用 S3 作为专业版的服务器，内部对象不会被损坏。

Seafile 服务器包含了 `seaf-fsck` 工具来帮助你恢复这些毁坏的对象（类似于 `git-fsck` 工具）。这个工具将会进行如下三项工作：

1. 检查 Seafile 内部对象的完整性，并且删除毁坏的对象。
2. 恢复所有受影响的资料库到最近一致，可用的状态。
3. 导出数据库。

执行流程如下所示：

```
cd seaf-server-latest
./seaf-fsck.sh [--repair|-r] [repo_id_1 [repo_id_2 ...]]
```

`seaf-fsck` 有检查资料库完整性和修复损坏资料库两种运行模式。

## 检查资料库完整性

执行 `seaf-fsck.sh` 不加任何参数将以只读方式检查所有资料库的完整性。

```
cd seaf-server-latest
./seaf-fsck.sh
```

如果你想检查指定资料库的完整性，只需将要检查的资料库 ID 作为参数即可：

```
cd seaf-server-latest
./seaf-fsck.sh [library-id1] [library-id2] ...
```

运行输出如下:

```
[02/13/15 16:21:07] fsck.c(470): Running fsck for repo ca1a860d-
e1c1-4a52-8123-0bf9def8697f.
[02/13/15 16:21:07] fsck.c(413): Checking file system integrity
of repo fsck(ca1a860d)...
[02/13/15 16:21:07] fsck.c(35): Dir 9c09d937397b51e1283d68ee7590
cd9ce01fe4c9 is missing.
[02/13/15 16:21:07] fsck.c(200): Dir /bf/pk/(9c09d937) is corrup
ted.
[02/13/15 16:21:07] fsck.c(105): Block 36e3dd8757edeb97758b3b4d8
530a4a8a045d3cb is corrupted.
[02/13/15 16:21:07] fsck.c(178): File /bf/02.1.md(ef37e350) is c
orrupted.
[02/13/15 16:21:07] fsck.c(85): Block 650fb22495b0b199cff0f1e1eb
f036e548fcb95a is missing.
[02/13/15 16:21:07] fsck.c(178): File /01.2.md(4a73621f) is curr
rupted.
[02/13/15 16:21:07] fsck.c(514): Fsck finished for repo ca1a860d
.
```

被损坏的文件和目录将显示在输出的结果中。

有时，你会看到如下的输出结果：

```
[02/13/15 16:36:11] Commit 6259251e2b0dd9a8e99925ae6199cbf4c134e
c10 is missing
[02/13/15 16:36:11] fsck.c(476): Repo ca1a860d HEAD commit is co
rrupted, need to restore to an old version.
[02/13/15 16:36:11] fsck.c(314): Scanning available commits...
[02/13/15 16:36:11] fsck.c(376): Find available commit 1b26b13c(
created at 2015-02-13 16:10:21) for repo ca1a860d.
```

这意味着记录在数据库中的 "head commit"（当前资料库状态的标识）与数据目录中的记录不一致。这种情况下，**fsck** 会试着找出最近可用的一致状态，并检查其完整性。

建议: 如果你有很多资料库要检查，保存 **fsck** 的输出到日志文件中将有助于后面的进一步分析。

## 修复损坏的资料库

**fsck** 修复损坏的资料库有如下两步流程:

1. 如果记录在数据库中的资料库当前状态无法在数据目录中找出，**fsck** 将会在数据目录中找到最近可用状态。
2. 检查第一步中可用状态的完整性。如果文件或目录损坏，**fsck** 将会将其置空并报告损坏的路径，用户便可根据损坏的路径来进行恢复操作。

执行如下命令来修复所有资料库：

```
cd seafile-server-latest
./seaf-fsck.sh --repair
```

大多数情况下我们建议你首先以只读方式检查资料库的完整性，找出损坏的资料库后，执行如下命令来修复指定的资料库：

```
cd seafile-server-latest
./seaf-fsck.sh --repair [library-id1] [library-id2] ...
```

**seaf-fsck** 会自动将改资料库的所有同步客户端解除同步。用户需要重新同步该资料库。**seaf-fsck** 也会在资料库的历史中添加一个损坏文件和目录的列表，便于用户找到损坏的路径。

## 修复资料库的最佳方案

检查所有的资料库并且找出已损坏的资料库，系统管理员可以运行不带任何参数的 **seaf\_fsck.sh**, 并且将输出结果保存到日志文件中。在日志文件中搜索关键字 **"Fail"** 来查找已损坏的库。

当管理员发现损坏的资料库时，他/她应该运行带有 **"--repair"** 的 **seaf-fsck.sh**。修复资料库后，管理员应该通知用户从其他地方恢复文件。有两个方法：

- 通过Web界面上传损坏的文件或文件夹
- 如果该资料库已经同步到某台计算机，并且该计算机有已损坏文件的正确版本，则重新同步该计算机上的资料库将上传已损坏文件的正确版本到服务器上。

## 将资料库导出到文件系统

4.2.0版本以后，您可以使用**seaf-fsck**将资料库中的所有文件导出到外部文件系统(如Ext4)。改程序不依赖于 **seaf** 数据库。只要你有 **seaf**-data 目录，你可以随时将文件从**seaf**导出到外部文件系统。

命令语法是

```
cd seaf-server-latest
./seaf-fsck.sh --export top_export_path [library-id1] [library-id2] ...
```

参数 **top\_export\_path** 是放置导出文件的目录。每个资料库将导出为导出目录的子目录。如果不指定资料库的ID，将导出所有库。

目前只能导出未加密的资料库，加密资料库将被跳过。

# Seafile GC

Seafile 利用存储去重技术来减少存储资源的利用。简单来说，这包含如下两层含义：

- 不同版本的文件或许会共享一些数据块。
- 不同的资料库也或许会共享一些数据块。

运用这项技术之后，在你删除一个资料库时，会导致底层数据块不会被立即删除，因此 Seafile 服务器端没用的数据块将会增多。

通过运行垃圾回收程序，可以清理无用的数据块，释放无用数据块所占用的存储空间。

垃圾回收程序将会清理如下两种无用数据块：

1. 未被资料库所引用的数据块即数据块属于被删除的资料库。
2. 设置了历史长度限制的资料库的过期数据块。

如果使用社区版服务器，运行垃圾回收程序之前，请先在服务器端停掉 **Seafile** 程序。这是因为垃圾回收程序，会错误的删除刚刚写入 **Seafile** 的新的数据块。对于专业版，**3.1.11** 及之后的版本，支持在线垃圾回收即如果使用 **MySQL** 或 **PostgreSQL** 数据库，你不需要暂停 **Seafile** 程序来进行垃圾回收。

## 4.1.1 及之后的版本

从社区版 4.1.1 和专业版 4.1.0 开始，我们改善了垃圾回收的命令参数和执行结果输出。

## Dry-run 模式

如果仅为了查看有多少垃圾可以回收而不进行删除操作，用 **dry-run** 选项：

```
seaf-gc.sh --dry-run [repo-id1] [repo-id2] ...
```

运行输出如下所示：

```
[03/19/15 19:41:49] seafserv-gc.c(115): GC version 1 repo My Lib
rary(ffa57d93)
```

```
[03/19/15 19:41:49] gc-core.c(394): GC started. Total block number is 265.
[03/19/15 19:41:49] gc-core.c(75): GC index size is 1024 Byte.
[03/19/15 19:41:49] gc-core.c(408): Populating index.
[03/19/15 19:41:49] gc-core.c(262): Populating index for repo ffa57d93.
[03/19/15 19:41:49] gc-core.c(308): Traversed 5 commits, 265 blocks.
[03/19/15 19:41:49] gc-core.c(440): Scanning unused blocks.
[03/19/15 19:41:49] gc-core.c(472): GC finished. 265 blocks total, about 265 reachable blocks, 0 blocks can be removed.

[03/19/15 19:41:49] seafserv-gc.c(115): GC version 1 repo aa(f3d0a8d0)
[03/19/15 19:41:49] gc-core.c(394): GC started. Total block number is 5.
[03/19/15 19:41:49] gc-core.c(75): GC index size is 1024 Byte.
[03/19/15 19:41:49] gc-core.c(408): Populating index.
[03/19/15 19:41:49] gc-core.c(262): Populating index for repo f3d0a8d0.
[03/19/15 19:41:49] gc-core.c(308): Traversed 8 commits, 5 blocks.
[03/19/15 19:41:49] gc-core.c(264): Populating index for sub-repo 9217622a.
[03/19/15 19:41:49] gc-core.c(308): Traversed 4 commits, 4 blocks.
[03/19/15 19:41:49] gc-core.c(440): Scanning unused blocks.
[03/19/15 19:41:49] gc-core.c(472): GC finished. 5 blocks total, about 9 reachable blocks, 0 blocks can be removed.

[03/19/15 19:41:49] seafserv-gc.c(115): GC version 1 repo test2(e7d26d93)
[03/19/15 19:41:49] gc-core.c(394): GC started. Total block number is 507.
[03/19/15 19:41:49] gc-core.c(75): GC index size is 1024 Byte.
[03/19/15 19:41:49] gc-core.c(408): Populating index.
[03/19/15 19:41:49] gc-core.c(262): Populating index for repo e7d26d93.
[03/19/15 19:41:49] gc-core.c(308): Traversed 577 commits, 507 blocks.
[03/19/15 19:41:49] gc-core.c(440): Scanning unused blocks.
[03/19/15 19:41:49] gc-core.c(472): GC finished. 507 blocks total
```

```
1, about 507 reachable blocks, 0 blocks can be removed.
```

```
[03/19/15 19:41:50] seafserv-gc.c(124): === Repos deleted by use
rs ===
```

```
[03/19/15 19:41:50] seafserv-gc.c(145): === GC is finished ===
```

```
[03/19/15 19:41:50] Following repos have blocks to be removed:
repo-id1
repo-id2
repo-id3
```

如果在参数中指定资料库 ID，则程序只检查指定的资料库，否则所有的资料库将会被检查。

在程序输出的结尾，你会看到 "repos have blocks to be removed" 部分，这部分内容会列出含有可回收垃圾块的资料库的 ID，后续你可以运行程序不加 `--dry-run` 选项来回收这些资料库的垃圾数据块。

## 删除垃圾数据块

运行垃圾回收程序，不加 `--dry-run` 选项来删除垃圾数据块：

```
seaf-gc.sh [repo-id1] [repo-id2] ...
```

如果在参数中指定资料库 ID，则程序只检查和删除指定的资料库。

正如前面所说，有两种类型的垃圾数据块可被回收，有时仅删除第一类无用数据块（属于删除的资料库）便可达到回收的目的，这种情况下，垃圾回收程序将不会检查未被删除的资料库，加入 `-r` 选项便可实现这个功能：

```
seaf-gc.sh -r
```

**Seafile 4.1.1** 及之后的版本，被用户删除的资料库不会直接从系统中删除，它们会被转移到系统管理员界面的垃圾箱。垃圾箱中的资料库，只有在从垃圾箱中清除以后，它们的数据块才可被回收。

## 3.1.2 及之后版本

运行垃圾回收程序



```
./seaf-gc.sh run
```

程序结束之后，运行以下命令，检查是否误删了还在使用的数据块，如果误删，会显示警告信息。

```
./seaf-gc.sh verify
```

可以通过 `dry-run` 选项，设置在运行垃圾回收程序前，进行完整性检查

程序将会显示 所有的数据块数量 和 将要被删除的数据块数量

```
./seaf-gc.sh dry-run
```

如果资料库已损坏，因为无法判断数据块是否还在被其他资料库使用，所以垃圾回收程序将会停止运行。

可以通过 `force` 选项，强制删除已损坏资料库的数据。通过将已损坏资料库的数据块标记为“未使用”，来将其删除。

```
./seaf-gc.sh force
```

## 3.1.2 及之前版本

运行垃圾回收程序

```
cd seafiler-server-{version}/seafiler
export LD_LIBRARY_PATH=./lib:${LD_LIBRARY_PATH}
./bin/seafserv-gc -c ../../ccnet -d ../../seafiler-data
```

如果你[源码编译安装 Seafiler 服务器](#)，仅仅运行

```
seafserv-gc -c ../../ccnet -d ../../seafiler-data
```

当垃圾回收程序结束后，你也可以检查是否一些有用的数据块被错误的删除：

```
seafserv-gc -c ../../ccnet -d ../../seafiler-data --verify
```

## Seafile GC

如果一些有用的数据块丢失，它将会打印一些警告信息。

如果你想在真正删除一些数据块之前，做一些常规检查，可以使用--dry-run选项

```
seafserv-gc -c ../../ccnet -d ../../seafile-data --dry-run
```

这将会向你展示数据块总数量和将被删除数据块数量。

如果在服务器端一些库的元数据被毁坏，垃圾回收程序将会停止处理，因为它无法识别是否一个数据块被一些毁坏的资料库所使用。如果你不想保留毁坏库的数据块，可以运行垃圾回收程序并使用--ignore-errors或-i选项。

```
seafserv-gc -c ../../ccnet -d ../../seafile-data --ignore-errors
```

这将会屏蔽毁坏资料库的数据块为无用状态并删除掉它们。

## WebDAV和FUSE扩展

Seafile WebDAV和FUSE扩展使得Seafile能够很容易的与第三方应用协调工作。例如，你可以在IOS上通过WebDAV接口访问Seafile上的文件。

## WebDAV扩展

Seafile WebDAV Server(SeafDAV)在Seafiler Server 2.1.0版本中被加入。

在下面的维基中,我们假设你将Seafiler安装到 `/data/haiwen` 目录下。

## SeafDAV配置

SeafDAV配置文件是 `/data/haiwen/conf/seafdav.conf`。如果它还没有被创建,你可以自行创建它。

```
[WEBDAV]
```

```
默认值是false。改为true来使用SeafDAV server。
```

```
enabled = true
```

```
port = 8080
```

```
如果fastcgi将被使用则更改fastcgi的值为true。
```

```
fastcgi = false
```

```
如果你将seafdav部署到nginx/apache,你需要更改“share_name”的值。
```

```
share_name = /
```

每次配置文件被修改后,你需要重启Seafiler服务器使之生效。

```
./seafiler.sh restart
```

## 示例配置 1: No nginx/apache

你的WebDAV客户端将在地址 `http://example.com:8080` 访问WebDAV服务器。

```
[WEBDAV]
enabled = true
port = 8080
fastcgi = false
share_name = /
```

## 示例配置 2: With Nginx/Apache

你的WebDAV客户端将在地址 `http://example.com/seafdav` 访问WebDAV服务器。

```
[WEBDAV]
enabled = true
port = 8080
fastcgi = true
share_name = /seafdav
```

在上面的配置中，"`share_name`"的值被改为"`/seafdav`"，它是你指定给seafdav服务器的地址后缀。

## Nginx 无 HTTPS

相应的Nginx配置如下 (无 https):

```

 location /seafdav {
 fastcgi_pass 127.0.0.1:8080;
 fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
 fastcgi_param PATH_INFO
$fastcgi_script_name;

 fastcgi_param SERVER_PROTOCOL
$server_protocol;
 fastcgi_param QUERY_STRING
$query_string;
 fastcgi_param REQUEST_METHOD
$request_method;
 fastcgi_param CONTENT_TYPE
$content_type;
 fastcgi_param CONTENT_LENGTH
$content_length;
 fastcgi_param SERVER_ADDR $server_addr;
 fastcgi_param SERVER_PORT $server_port;
 fastcgi_param SERVER_NAME $server_name;

 client_max_body_size 0;

 access_log
/var/log/nginx/seafdav.access.log;
 error_log /var/log/nginx/seafdav.error.log;
 }

```

## Nginx 有 HTTPS

Nginx配置为https:

```

 location /seafdav {
 fastcgi_pass 127.0.0.1:8080;
 fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
 fastcgi_param PATH_INFO
$fastcgi_script_name;

 fastcgi_param SERVER_PROTOCOL
$server_protocol;
 fastcgi_param QUERY_STRING
$query_string;
 fastcgi_param REQUEST_METHOD
$request_method;
 fastcgi_param CONTENT_TYPE
$content_type;
 fastcgi_param CONTENT_LENGTH
$content_length;
 fastcgi_param SERVER_ADDR $server_addr;
 fastcgi_param SERVER_PORT $server_port;
 fastcgi_param SERVER_NAME $server_name;

 client_max_body_size 0;

 fastcgi_param HTTPS on;

 access_log
/var/log/nginx/seafdav.access.log;
 error_log /var/log/nginx/seafdav.error.log;
 }

```

## Apache

首先编辑 `apache2.conf` 文件, 添加如下这行到文件结尾(或者根据你的Linux发行版将其添加到 `httpd.conf` ):

```
FastCGIExternalServer /var/www/seafdav.fcgi -host
127.0.0.1:8080
```

注意, `/var/www/seafdav.fcgi` 仅仅只是一个占位符, 实际在你的系统并不需要  
有此文件。

第二, 修改Apache配置文件 (site-enabled/000-default):

## Apache 无 HTTPS

根据你的Apache配置当你[将要部署 Seafile 和 Apache]已经部署 Seafile 和  
Apache], 加入Seafdav的相关配置:



```
ServerName www.myseafile.com
 DocumentRoot /var/www
 Alias /media /home/user/haiwen/seafile-
server/seahub/media

 RewriteEngine On

 #
 # seafile fileserver
 #
 ProxyPass /seafhttp http://127.0.0.1:8082
 ProxyPassReverse /seafhttp http://127.0.0.1:8082
 RewriteRule ^/seafhttp - [QSA,L]

 #
 # seafile webdav
 #
 RewriteCond %{HTTP:Authorization} (.+)
 RewriteRule ^(/seafdav.*)$ /seafdav.fcgi$1
[QSA,L,e=HTTP_AUTHORIZATION:%1]
 RewriteRule ^(/seafdav.*)$ /seafdav.fcgi$1 [QSA,L]

 #
 # seahub
 #
 RewriteRule ^/(media.*)$ /$1 [QSA,L,PT]
 RewriteCond %{REQUEST_FILENAME} !-f
 RewriteRule ^(.*)$ /seahub.fcgi$1
[QSA,L,E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
```

## Apache 有 HTTPS

根据你的apache配置当你配置Seafile网站和Apache并启用Https, 加入seafdav的相关配置:

```
ServerName www.myseafile.com
 DocumentRoot /var/www
 Alias /media /home/user/haiwen/seafile-
server/seahub/media

 SSLEngine On
 SSLCertificateFile /etc/ssl/cacert.pem
 SSLCertificateKeyFile /etc/ssl/privkey.pem

 RewriteEngine On

 #
 # seafile fileserver
 #
 ProxyPass /seafhttp http://127.0.0.1:8082
 ProxyPassReverse /seafhttp http://127.0.0.1:8082
 RewriteRule ^/seafhttp - [QSA,L]

 #
 # seafile webdav
 #
 RewriteCond %{HTTP:Authorization} (.+)
 RewriteRule ^(/seafdav.*)$ /seafdav.fcgi$1
[QSA,L,e=HTTP_AUTHORIZATION:%1]
 RewriteRule ^(/seafdav.*)$ /seafdav.fcgi$1 [QSA,L]

 #
 # seahub
 #
 RewriteRule ^/(media.*)$ /$1 [QSA,L,PT]
 RewriteCond %{REQUEST_FILENAME} !-f
 RewriteRule ^(.*)$ /seahub.fcgi$1
[QSA,L,E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
```

## 关于客户端的注意事项

### Windows

在Windows平台，我们推荐使用webdav客户端软件例如Cyberduck或BitKinex。webdav对于Windows浏览器的支持实现并不是十分可用，因为：

Windows 浏览器需要利用HTTP数字认证。但是由于Seafile在服务器端不存储纯文本密码，所以它不支持这个特性。HTTP基本认证只被HTTPS支持（这是合理的）。但是浏览器不支持自我签署的证书。

结论就是如果你有一个合法的ssl证书，你应该能通过Windows浏览器来访问seafdav。否则你应该使用客户端软件。Windows XP被声明不支持HTTPS webdav。

### Linux

在Linux平台你有更多的选择。你可以利用文件管理器例如Nautilus来连接webdav服务器，或者在命令行使用davfs2。

使用davfs2

```
sudo apt-get install davfs2
sudo mount -t davfs -o uid= https://example.com/seafdav
/media/seafdav/
```

-o选项设置挂载目录的拥有者为，使得非root用户拥有可写权限。

我们建议对于davfs2，禁用锁操作。你需要编辑/etc/davfs2/davfs2.conf

```
use_locks 0
```

### Mac OS X

Finder对于WebDAV的支持不稳定而且较慢. 所以我们建议使用webdav客户端软件如Cyberduck.

## 常见问题

### 客户端无法连接seafdav服务器

默认, seafdav是未被启用的。检查你是否在 `seafdav.conf` 中设置 `enabled = true`。如果没有, 更改配置文件并重启seafle服务器。

### 客户端得到"**Error: 404 Not Found**"错误

如果你将SeafDAV部署在Nginx/Apache, 请确保像上面的配置文件一样更改 `share_name` 的值。重启Seafle服务器后重新测试。

# FUSE 扩展

在Seafile系统上文件被分割成数据块，这意味着在你的Seafile服务器上存储的并不是完整的文件而是数据块。这种设计能够方便有效的运用数据去重技术。

然而，有时系统管理员想要直接访问服务器上的文件，你可以使用seaf-fuse来做到这点。

Seaf-fuse 是一种FUSE虚拟文件系统的实现。一句话来说就是，它挂载所有的Seafile文件到一个目录（它被称为“挂载点”），所以你可以像访问服务器上的正常目录一样来访问由Seafile服务器管理的所有文件。

注意：

- 加密的目录不可以被seaf-fuse来访问。
- Seaf-fuse的当前实现是只读访问，这意味着你不能通过挂载的目录来修改文件。
- 对于debian/centos系统，你需要在“fuse”组才有权限来挂载一个FUSE目录。

## 如何启动seaf-fuse

假设你想挂载到 `/data/seaf-fuse` 。

创建一个目录作为挂载点

```
mkdir -p /data/seaf-fuse
```

## 用脚本来启动seaf-fuse

注意： 在启动seaf-fuse之前，你应该已经通过执行 `./seaf.sh start` 启动好Seafile服务器。

```
./seaf-fuse.sh start /data/seaf-fuse
```

## 停止seaf-fuse

```
./seaf-fuse.sh stop
```

## 挂载目录的内容

### 顶层目录

现在你可以列出 `/data/seafile-fuse` 目录的内容

```
$ ls -lhp /data/seafile-fuse

drwxr-xr-x 2 root root 4.0K Jan 1 1970 abc@abc.com/
drwxr-xr-x 2 root root 4.0K Jan 1 1970 foo@foo.com/
drwxr-xr-x 2 root root 4.0K Jan 1 1970 plus@plus.com/
drwxr-xr-x 2 root root 4.0K Jan 1 1970 sharp@sharp.com/
drwxr-xr-x 2 root root 4.0K Jan 1 1970 test@test.com/
```

- 顶层目录包含许多子目录，每个子目录对应一个用户
- 文件和目录的时间戳不会被保存

### 每个用户的目录

```
$ ls -lhp /data/seafile-fuse/abc@abc.com

drwxr-xr-x 2 root root 924 Jan 1 1970 5403ac56-5552-4e31-a4f1-1de4eb889a5f_Photos/
drwxr-xr-x 2 root root 1.6K Jan 1 1970 a09ab9fc-7bd0-49f1-929d-6abeb8491397_My Notes/
```

从上面的列表可以看出，在用户目录下有一些子目录，每个子目录代表此用户的一个资料库，并且以“{库id}-{库名字}”的格式来命名。

### 资料库的目录

```
$ ls -lhp /data/seafile-fuse/abc@abc.com/5403ac56-5552-4e31-a4f1-1de4eb889a5f_Photos/

-rw-r--r-- 1 root root 501K Jan 1 1970 image.png
-rw-r--r-- 1 root root 501K Jan 1 1970 sample.jpg
```

## 如果出现"**Permission denied**"的错误

如果你运行 `./seaf-fuse.sh start` 时，遇到"**Permission denied**"的错误信息，很有可能你没有在“fuse用户组”解决方法：

- 把你的用户加到fuse组

```
sudo usermod -a -G fuse
```

- 退出shell重新登陆
- 现在试着再一次执行 `./seaf-fuse.sh start <path>` 。

## 安全和审计

### 安全特性

- [安全特性](#)

### 访问日志和审计

- [访问日志和审计](#)



## 安全机制

### 客户端和服务端间的通信加密

Seafile 在服务器配置了 HTTPS 后，客户端会自动使用 HTTPS 协议和服务端通信。

### 加密资料库如何工作？

当你创建一个加密资料库，你将为其提供一个密码。所有资料库中的数据在上传到服务器之前都将用密码进行加密。

加密流程：

1. 生成一个32字节长的加密的强随机数。它将被用作文件加密键（“文件键”）。
2. 用用户提供的密码对文件键进行加密。我们首先用PBKDF2算法从密码中获取到一个键/值对，然后用AES 256/CBC来加密文件键，所得结果被称之为“加密的文件键”。加密的文件键将会被发送到服务器并保存下来。当你需要访问那部分数据，你可以从加密的文件键中解密出文件键。
3. 所有的文件数据都将用AES 256/CBC加密的文件键进行加密。我们用PBKDF2算法从文件键中获取键/值对。加密完成后，数据将会被传送到服务器端。

上述加密过程即可在桌面客户端执行也可在网站浏览器中执行。浏览器端加密功能可在服务器端使用。当你从加密的资料库中上传和下载时，如下过程将会发生：

- 服务器端发回加密的数据，浏览器将会在客户端用JavaScript解密它们。
- 浏览器在客户端用JavaScript加密后，将加密后的数据发回服务器。服务器端直接保存加密后的结果。

在上述过程中，你的密码将不会在服务器端传输。

当你同步一个加密资料库到桌面客户端或者在网站浏览器中浏览一个资料库，桌面客户端/浏览器需要确认你的密码。当你创建一个资料库，一个“魔力标志”将会在密码和资料库id中获得。这个标志和资料库一起存储到服务器端。客户端用这个标志检查你的密码是否正确在你同步和浏览资料库之前。魔力标志是通过PBKDF2算法经过1000次迭代产生，所以它将非常安全抵抗蛮力破解。

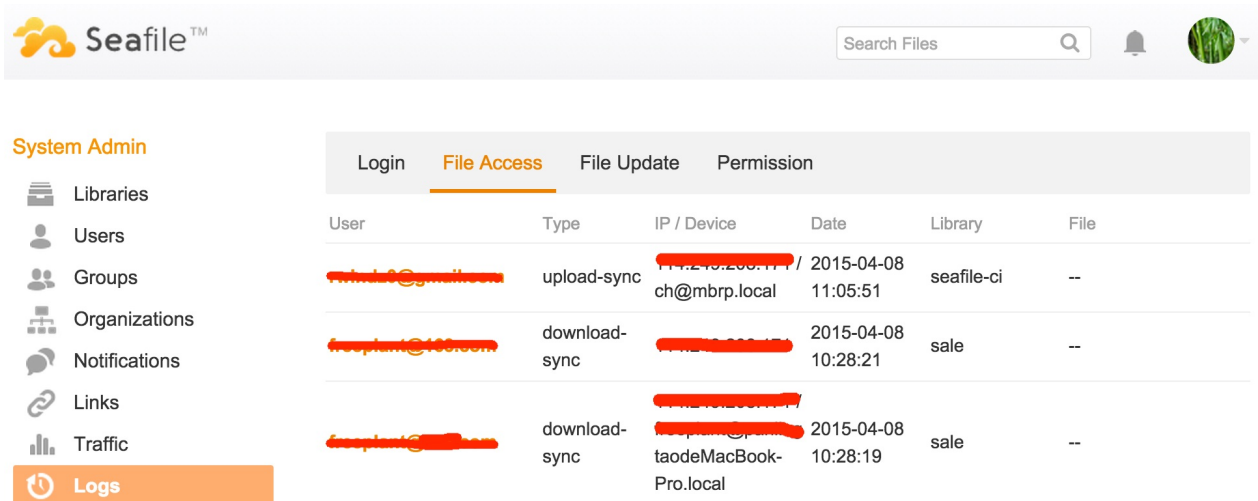
## 安全特性

为了最大安全性，纯文本的密码也不会保存在客户端。客户端只保存从“文件键”获得的键/值对，它用来解密数据。所以如果你忘记密码，你将不能恢复和访问服务器端的数据。

## 访问记录和审计

Seafile 企业版在管理员界面中提供了四类日志：

- 登录日志
- 文件访问日志
- 文件更新日志
- 权限更改日志



The screenshot shows the Seafile System Admin interface. On the left is a sidebar with navigation links: Libraries, Users, Groups, Organizations, Notifications, Links, Traffic, and Logs (highlighted in orange). The main content area has a tabbed interface with four tabs: Login, File Access (selected), File Update, and Permission. Below the File Access tab is a table with the following columns: User, Type, IP / Device, Date, Library, and File. The table contains three entries:

| User       | Type          | IP / Device                       | Date                | Library    | File |
|------------|---------------|-----------------------------------|---------------------|------------|------|
| [redacted] | upload-sync   | 11.2.43.200 / ch@mbrp.local       | 2015-04-08 11:05:51 | seafile-ci | --   |
| [redacted] | download-sync | [redacted]                        | 2015-04-08 10:28:21 | sale       | --   |
| [redacted] | download-sync | [redacted] taodeMacBook-Pro.local | 2015-04-08 10:28:19 | sale       | --   |

日志功能默认是关闭的，以便不产生大量的数据库条目。参考文档 [config options for pro edition](#) 来开启这个功能。

# 开发文档

请参考[英文版手册](#)